

Systems Bioinformatics

An Engineering Case-Based Approach

DISCLAIMER OF WARRANTY

The technical descriptions, procedures, and computer programs in this book have been developed with the greatest of care and they have been useful to the authors in a broad range of applications; however, they are provided as is, without warranty of any kind. Artech House, Inc., and the authors and editors of the book titled *Systems Bioinformatics: An Engineering Case-Based Approach* make no warranties, express or implied, that the equations, programs, and procedures in this book or its associated software are free of error, or are consistent with any particular standard of merchantability. They should not be relied upon for solving a problem whose incorrect solution could result in injury to a person or loss of property. Any use of the programs or procedures in such a manner is at the user's own risk. The editors, authors, and publisher disclaim all liability for direct, incidental, or consequent damages resulting from use of the programs or procedures in this book or the associated software.

The Artech House Bioinformatics & Biomedical Imaging Series
Steven Wong, Harvard Medical School, and Guang-Zhong Yang, Imperial College, Series Editors

For a listing of recent related Artech House titles, please turn to the back of this book.

Systems Bioinformatics

An Engineering Case-Based Approach

Gil Alterovitz
Marco F. Ramoni
Editors



**ARTECH
HOUSE**

BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data

A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library.

ISBN 13: 978-1-59693-124-4

Cover design by Igor Valdman

© 2007 ARTECH HOUSE, INC.

685 Canton Street

Norwood, MA 02062

All rights reserved. Printed and bound in the United States of America. No part of this book may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Artech House cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

10 9 8 7 6 5 4 3 2 1

To our parents

Contents

Preface	xv
PART I	
Introduction: Molecular and Cellular Biology	1
CHAPTER 1	
Molecular and Cellular Biology: An Engineering Perspective	3
1.1 Cellular Structures and Functions	3
1.2 Introduction to Information Handling in Cells	4
1.3 The Importance and Diversity of Proteins	5
1.4 DNA Replication: Copying the Code	6
1.5 Transcription: Sending a Messenger	7
1.6 Translation: Protein Synthesis	9
1.7 Control of Gene Expression	11
1.8 Genetic Engineering	12
1.9 Summary	13
CHAPTER 2	
Proteomics: From Genome to Proteome	15
2.1 Defining the Proteome	15
2.1.1 From Genes to Proteins	15
2.1.2 What Is Proteomics?	17
2.1.3 Functional Proteomics	18
2.2 Building Gene Collections for Functional Proteomics Approaches	18
2.2.1 Selection of Target Genes for a Cloning Project	21
2.2.2 Clone Production	25
2.2.3 Sequencing and Analysis	32
2.2.4 Clone Maintenance and Distribution	34
2.3 Use of Clones in Functional Proteomics Approaches	35
2.3.1 High-Throughput Protein Production	36
2.3.2 Protein Arrays	38
2.3.3 Cell-Based Functional Proteomic Assays	39

PART II

Analysis: Signal Processing	47
-----------------------------	----

CHAPTER 3

Introduction to Biological Signal Processing at the Cell Level	49
3.1 Introduction to Fundamental Signal Processing Concepts	51
3.1.1 Signals	51
3.1.2 Systems	54
3.1.3 Random Processes and Spectral Analysis	57
3.2 Signal Detection and Estimation	59
3.2.1 DNA Sequencing	60
3.2.2 Gene Identification	67
3.2.3 Protein Hotspots Identification	71
3.3 System Identification and Analysis	74
3.3.1 Gene Regulation Systems	77
3.3.2 Protein Signaling Systems	84
3.4 Conclusion	93

CHAPTER 4

Signal Processing Methods for Mass Spectrometry	101
4.1 Introduction	101
4.1.1 Data Acquisition Methods	102
4.1.2 History of Ionization Techniques	102
4.1.3 Sample Preparation	103
4.1.4 Ionization	103
4.1.5 Separation of Ions by Mass and Charge	103
4.1.6 Detection of Ions and Recorded Data	104
4.1.7 Data Preprocessing	104
4.1.8 Example Data	105
4.2 Signal Resampling	105
4.2.1 Algorithm Explanation and Discussion	106
4.2.2 Example Demonstrating Down Sampling	107
4.3 Correcting the Background	109
4.3.1 Algorithm Explanation and Discussion	109
4.3.2 Example Demonstrating Baseline Subtraction	111
4.4 Aligning Mass/Charge Values	112
4.4.1 Algorithm Explanation and Discussion	113
4.4.2 Example Demonstrating Aligning Mass/Charge Values	114
4.5 Normalizing Relative Intensity	116
4.5.1 Example Demonstrating Intensity Normalization	116
4.6 Smoothing Noise	119
4.6.1 Lowess Filter Smoothing	120
4.6.2 Savitzky and Golay Filter Smoothing	121
4.6.3 Example Demonstrating Noise Smoothing	121
4.7 Identifying Ion Peaks	122

PART III

Analysis: Control and Systems	125
-------------------------------	-----

CHAPTER 5

Control and Systems Fundamentals	127
----------------------------------	-----

5.1	Introduction	127
5.2	Review of Fundamental Concepts in Control and Systems Theory	128
5.2.1	Discrete-Time Dynamical Systems	132
5.3	Control Theory in Systems Biology	133
5.4	Reverse Engineering Cellular Networks	135
5.5	Gene Networks	137
5.5.1	Boolean Networks	139
5.5.2	Dynamic Bayesian Networks	143
5.6	Conclusion	147

CHAPTER 6

Modeling Cellular Networks	151
----------------------------	-----

6.1	Introduction	151
6.2	Construction and Analysis of Kinetic Models	153
6.2.1	Parameter Estimation and Modeling Resources	153
6.2.2	A Modular Approach to Model Formulation	154
6.2.3	Basic Kinetics	156
6.2.4	Deterministic Models	158
6.2.5	Cellular Noise and Stochastic Methods	158
6.2.6	System Analysis Techniques	161
6.3	Case Studies	164
6.3.1	Expression of a Single Gene	164
6.3.2	A Phosphorylation-Dephosphorylation Cycle	166
6.3.3	A Synthetic Population Control Circuit	168
6.4	Conclusion	172

PART IV

Analysis: Probabilistic Data Networks and Communications	179
--	-----

CHAPTER 7

Topological Analysis of Biomolecular Networks	181
---	-----

7.1	Cellular Networks	181
7.1.1	Genetic Regulation Networks	182
7.1.2	Protein-Protein Interaction Networks	184
7.1.3	Metabolic Regulation Networks	185
7.1.4	The Scale-Free Property: A Network Characteristic	186
7.2	The Topology of Cellular Networks	189
7.2.1	Network Motifs in Genetic Regulation Networks	189
7.2.2	Topological Characterization of Protein Networks	191
7.2.3	Topology of Metabolic Networks	192

7.2.4	Adjacency Matrices	196
7.2.5	Hubs	196
7.2.6	Reachability	197
7.3	Gene Ontology and Functional Clustering of Essential Genes	198
7.4	Conclusion and Future Avenues	201

CHAPTER 8

Bayesian Networks for Genetic Analysis		205
8.1	Introduction	205
8.2	Elements of Population Genetics	206
8.3	Bayesian Networks	210
8.3.1	Representation	210
8.3.1	Learning	213
8.3.3	Reasoning	217
8.3.4	Validation and Inference	219
8.3.5	Risk Prediction	219
8.4	Two Applications	221
8.4.1	Stroke Risk in Sickle Cell Anemia Subjects	221
8.4.2	Network Representation of a Complex Trait	221
8.5	Conclusion	224

PART V

Design: Synthetic Biology		229
---------------------------	--	-----

CHAPTER 9

Fundamentals of Design for Synthetic Biology		231
9.1	Overview	231
9.2	Circuits	232
9.2.1	Riboregulators	234
9.2.2	Feedback Loops	235
9.2.3	Toggle Switches	236
9.2.4	Logic Gates	236
9.2.5	Oscillators	236
9.3	Multicellular Systems	236
9.4	Challenges	238
9.4.1	Standardization	238
9.4.2	Stochasticity	238
9.4.3	Directed Evolution	239
9.4.4	Random and Targeted Mutagenesis and Recombination	239
9.4.5	System Interface	240
9.4.6	Kinetics	240
9.5	Conclusion	240

CHAPTER 10

BioJADE: Designing and Building Synthetic Biological Systems from Parts	243
10.1 Introduction	243
10.2 Fundamentals of BioJADE and BioBricks Construction	243
10.2.1 Inspiration	243
10.2.2 The BioBricks Standard	244
10.2.3 BioBrick Definition	244
10.2.4 The Abstraction Barrier	245
10.3 Representing Parts	246
10.3.1 Parts Data Model	247
10.4 BioJADE Architecture	248
10.4.1 Aspects	248
10.4.2 Schematic	249
10.4.3 Functional Network Aspect	250
10.4.4 DNA Aspect	250
10.4.5 Icon Aspect	251
10.4.6 Part Repositories	251
10.5 Using BioJADE, an Example: The Repressilator	251
10.6 Simulations	254
10.6.1 D-FLUX	254
10.6.2 Stochastirator	255
10.6.3 Tabasco	255
10.6.4 Generating the Simulation	256
10.7 The Reality Check	257
10.7.1 Biological Circuit Design Cannot Be as Easy as VLSI Design	257
10.7.2 Bugs Fight Back	257
10.8 Next Steps	258
10.8.1 Simulations	258
10.8.2 Parts	259
10.8.3 Designing Systems	259
10.8.4 Measurement	259

CHAPTER 11

Applied Cellular Engineering	263
11.1 Introduction	263
11.1.1 Biological Systems Engineering	263
11.1.2 Cellular Catalytic Machinery	265
11.1.3 Early Engineering Successes	265
11.2 Engineering Tools	266
11.2.1 Network Models and Analysis	266
11.2.2 Experimental Methods	271
11.3 Case Study: Production of 1,3-Propanediol in <i>E. coli</i>	277
11.4 Frontiers	277
11.5 Conclusion	278

PART VI

Integration: Applying Biology's Designs and Principles in Engineering	283
---	-----

CHAPTER 12

The Three Faces of DNA/RNA Sequence Hybridization	285
12.1 Introduction	285
12.2 A Short Introduction to DNA/RNA Sequence Hybridization and Self-Hybridization	286
12.3 DNA/RNA Sequence Hybridization: A Biological Point of View	289
12.3.1 Functional RNA Molecules	289
12.3.2 Gene Silencing and RNA Interference	291
12.3.3 RNA Editing and Re-encoding	291
12.3.4 Fragile DNA Regions and Secondary Structures	293
12.4 DNA/RNA Sequence Hybridization: A Technological Point of View	294
12.4.1 DNA Computers	294
12.4.2 DNA Microarrays	298
12.4.3 DNA Cryptography	299
12.4.4 DNA/RNA-Aided Nanoparticle Assembly	300
12.5 DNA/RNA Sequence Hybridization: A Coding-Theoretic Point of View	301
12.5.1 DNA Codes	301
12.5.2 DNA Microarrays	307
12.5.3 Enumerating RNA Motifs	310
12.6 Conclusion	313

CHAPTER 13

Application of Biomolecular Computing to Breakthroughs in Cryptography	319
13.1 Introduction	319
13.2 Introduction of DNA Background	321
13.2.1 DNA Manipulations	321
13.2.3 Comparisons of Various Famous DNA Models	322
13.3 Factoring the Product of Two Large Prime Numbers	323
13.3.1 Introduction to the RSA Public-Key Cryptosystem	323
13.3.2 Solution Space of DNA Strands for Every Unsigned Integer	323
13.3.3 Construction of the Product for Two Large Prime Numbers	324
13.3.4 Construction of a Parallel Comparator	325
13.3.5 Construction of a Parallel One-Bit Subtractor	327
13.3.6 Construction of a Binary Parallel Subtractor	330
13.3.7 Construction of a Binary Parallel Divider	331
13.3.8 Finding Two Large Prime Numbers	334
13.3.9 Breaking the RSA Public-Key Cryptosystem	335
13.3.10 The Complexity of Algorithm 1	336
13.4 Conclusion	336

CHAPTER 14

Chemotaxis: Learning Navigation and Source Localization Strategies from Biology's Engineered Designs	341
14.1 Introduction	341
14.2 Bacterial Chemotaxis Principles	342
14.3 Mathematical Description of a Random Walk	344
14.4 Chemotaxis-Based Algorithms for Diffusive Environments	345
14.4.1 Single-Node Biased Random Walk and Receptor Cooperation	346
14.4.2 Multinode Biased Random Walks for Source Tracking	347
14.4.3 Multichemoreceptor Cooperation for Gradient Tracking	350
14.5 Performance Comparison of the Chemotaxis Algorithms	360
14.6 Summary	361
Systems Bioinformatics: Trends and Conclusions	365
Appendix: Contributing Authors and Contact Information	367
About the Editors	371
Index	373

Preface

The high-throughput nature of bioinformatics and system biology has made traditional biological methods, which tend to focus one or two molecules at a time, obsolete. It has made engineering and problem solving skills essential to attack the resulting complex, multiscale problems. Certain technologies, such as robotics automation, microfabrication, control, and signal processing are particularly amenable to the engineering expertise of electrical and other engineering disciplines.

This book presents a quantitative, case-based approach to the intersection of systems biology and bioinformatics: systems bioinformatics. This book takes the reader through the field's challenges: from the lab bench to data analysis and modeling. It has a different perspective than that of other books on systems biology and bioinformatics in that it presents a case-based approach using an engineering perspective. Each part starts with text on the engineering fundamentals and then focuses on an application via systems bioinformatics. The book is the result of an international effort across the world, involving nearly twenty institutions across five countries.

The material is designed to match ideas that engineering students are familiar with, such as analysis, design, and reverse engineering. These principles are demonstrated and explored within the context of the functionality in living systems. Thus, this book provides a systems approach to looking at biological processes, a core principle of the evolving umbrella of the intersection of systems biology and bioinformatics. It allows for the depth needed for engineering studies, while at the same time providing the underlying biological context.

Some of the engineering areas featured in this book include digital signal processing (Part II), control systems (Part III), communications (Part IV), and chemical engineering (Part V). Part VI deals with the idea of reverse engineering, which a majority of engineers can relate to. This book's distinctive engineering-oriented coverage makes the material more intuitive for a technical audience.

Through teaching at Harvard, MIT, and Boston University, the editors have found that students and professionals also gain a better understanding of their *own* engineering fields through learning about how their field's core concepts apply to other disciplines. Upon seeing the need for engineers in the nascent fields of bioinformatics and proteomics, the editors initiated two related courses at Harvard/MIT: HST.480/6.092 (Bioinformatics and Proteomics: An Engineering-Based Problem Solving Approach). The teaching approach used in those courses was subsequently published (Alterovitz, G., and M. F. Ramoni, "Bioinformatics and proteomics: an engineering-based problem solving approach," *IEEE Trans. on Education*, 2007). This book was developed as a result of these courses that the editors codirected at

the Massachusetts Institute of Technology (MIT) Electrical Engineering and Computer Science Department and Harvard/MIT Health Science and Technology (with Professor Manolis Kellis of MIT).

Like the courses it originated from, this book targets upper level undergraduate and first year graduate students in engineering disciplines. It does not try to cover every subfield of bioinformatics; rather, it seeks to teach concepts and ways of thinking about biological problems using an engineering approach. To do this, it is organized by engineering concepts, and presents cases in biology for in-depth exploration. Thus, this book is an excellent stand-alone text for an introductory/motivational seminar or course on the subject. It can also serve in juxtaposition to a more classically organized text—which covers the breadth of bioinformatics—by adding in-depth cases for study. Last, it can serve as a complementary text to traditional texts, which are often organized by biological concepts. By teaching bioinformatics from multiple perspectives, the editors have found that students gain a deeper understanding of the fundamental concepts.

The book has the recommended co- or prerequisites of Signals and Systems (e.g., 6.003 at MIT), Probabilistic Systems Analysis and Applied Probability (e.g., MIT 6.041/6.431), and Introductory Biology (Molecular) (e.g., MIT 7.012). For those who have not had one or more of the above classes, a couple of review sessions may be useful.

Some of the course materials and methodologies from the HST.480/6.092 courses (now in this book) were also subsequently used in HST.512 Genomic Medicine at Harvard Medical School, 6.872/HST 950 Biomedical Computing at MIT, and BS771 Design and Analysis of Microarray Experiments at Boston University. In addition, the 6.092 course was published on online for the public via MIT's Open Courseware initiative (<http://ocw.mit.edu>), a potentially useful online resource for readers.

The text is divided into six parts. Contrary to most bioinformatics books that present material based on biological concepts, this book's parts are categorized based on fundamental engineering concepts and applications, in a manner consistent with its engineering-oriented approach.

In Part I, the fundamental biology is introduced from an engineering perspective. The first chapter presents the needed molecular and cellular biology background and can be treated within a review session if the course includes a prerequisite biology course similar to MIT's 7.012 "Introductory Biology" (Molecular). A number of engineering analogies are presented to facilitate presentation of the material. In the second chapter, the book moves from the genomics to proteomics—looking at ways that engineering and automation can be used to explore genes and proteins in a parallel, high-throughput manner.

Parts II through IV focus on engineering analysis methods. Part II starts with signal processing methods. Chapter 3 introduces biological signal processing with applications, while Chapter 4 focuses on a case study in mass spectrometry. Part III discusses controls and systems. Chapter 5 introduces the fundamentals and applications in gene regulation. Chapter 6 focuses on modeling cellular circuits. In Part IV, probabilistic data networks and communications are covered. Chapter 7 dis-

cusses topologies of cellular networks and how some biological properties can be ascertained solely from network connectivity. The final chapter of this part, Chapter 8, introduces and expands on the use of Bayesian networks to link genetic information (single nucleotide polymorphisms, or SNPs) to human disease.

Parts V and VI switch from discussing analysis to tackling issues in design. After introducing the area of synthetic biology in Chapter 9, Part V goes on to look at computer-aided design (CAD) tools adapted from circuit design to biomolecular circuitry design in Chapter 10. Next, a case study with a chemical engineering industrial perspective is presented on applying cellular engineering to perturb cellular pathways. The final part, Part VI, looks at how biological designs and principles can be applied back to engineering. In Chapter 12, the biology of sequence hybridization is discussed along with its various applications to engineering, ranging from DNA-based computers to nanoparticle assembly.

In Chapter 13, it is shown how massive parallelization via DNA computing can be used to break encryption algorithms previously thought to be secure. Finally, Chapter 14 examines how navigation and source localization strategies can be inspired by biological designs involving chemotaxis. The book concludes by summarizing the field and looking at future avenues of research in this area. For those interested in additional resources, source code, and related materials, the book's Internet site can be accessed under artechhouse.com.

Because this work has been an international effort, there are many people whose contributions were critical to its publication. The editors would like to thank the editors at Artech House—particularly acquisitions editor Wayne Yuhasz, who invited us to write this book and worked hard with us to complete it on a tight schedule—and Barbara Lovenvirth, who helped in the manuscript review process. The editors would like to say thank you to 6.092 co-course director Manolis Kellis, Assistant Professor at the MIT Electrical Engineering and Computer Science Department; to Prof. Isaac Kohane at Harvard Medical School; to the Harvard Partners Center for Genetics and Genomics; and to the faculty and student members of the Harvard/MIT Health Science and Technology Division's Graduate Committee as well as the Electrical Engineering and Computer Science Division, especially Anne Hunter, for their support of the HST 480/6.092 courses.

The editors would like to thank the contributing authors to the text: Gregory Crowther, Catherine Speake, Alicia McBride, and Mary Lidstrom (Chapter 1); Stephanie Mohr, Yanhui Hu, and Joshua LaBaer (Chapter 2); Maya Said (Chapter 3); Peter Monchamp, Lucio Cetto, Jane Zhang, and Rob Henson (Chapter 4); Fulvia Ferrazzi and Riccardo Bellazzi (Chapter 5); Tae Jun Lee, Chee Meng Tan, Dennis Tu, and Lingchong You (Chapter 6); Vinayak Muralidhar and Gabor Szabo (Chapter 7); Paola Sebastiani and Maria Abad-Grau (Chapter 8); Cody Wood (Chapter 9); Jonathan Goler and Tom Knight (Chapter 10); Brian Baynes and William Blake (Chapter 11); Olgica Milenkovic (Chapter 12); Michael Shan-Hui Ho, Weng-Long Chang, and Minyi Guo (Chapter 13); and Gail Rosen and Paul Hasler (Chapter 14).

Additionally, the editors would like to acknowledge Ehsan Afkhami, now at Mathworks, for his contributions early in the book development process. Thank you as well to the following people who helped in reviewing and editing the

manuscript: Mamta Mohan, Amy Berninger, Victor Wong, and Dmitriy Sonkin. Finally, special thanks to the anonymous reviewers of the book proposal and draft.

Gil Alterovitz
Marco F. Ramoni
Editors
Boston, Massachusetts
February 2007

PART I

Introduction: Molecular and Cellular Biology

Molecular and Cellular Biology: An Engineering Perspective

Gregory J. Crowther, Catherine C. Speake,
Alicia A. McBride, and Mary E. Lidstrom

1.1 Cellular Structures and Functions

Biology is the study of living things, but what does it mean to say that something is alive? One approach is to define living organisms according to the core functions that distinguish them from nonliving systems. Key functions of essentially all organisms include intake of nutrients, use of these nutrients for growth and repair, excretion of wastes, self-reproduction, and the ability to sense and respond to environmental stimuli. Any single function listed here does not distinguish perfectly between living and nonliving things; for example, sterile organisms cannot reproduce themselves, whereas computer viruses can. However, with few exceptions, living organisms can perform all of the above functions, whereas nonliving things cannot.

All organisms consist of one or more cells, the basic structural unit of an organism. Cells are bound by a semipermeable membrane made predominantly of lipids and proteins; internally, they contain a variety of parts specialized for different functions (Table 1.1). Cells can be classified as eukaryotic (having a nucleus) or prokaryotic (not having a nucleus); bacteria are prokaryotic cells, which tend to be smaller and simpler than eukaryotic cells, the kind found in plants, animals, fungi, and protists. The interior of eukaryotic cells is divided into membrane-bound compartments called organelles. Examples of organelles include the nucleus, where the cell's DNA (deoxyribonucleic acid) is stored; mitochondria, which produce ATP (adenosine triphosphate) to be used in energy-requiring cellular processes; chloroplasts, which capture light energy and convert it to usable chemical energy; the endoplasmic reticulum, whose surface contains ribosomes for making proteins; Golgi complexes, which attach sugars to newly synthesized proteins before shipping them off to other parts of the cell; and lysosomes, which digest old or unwanted materials. Details about these and many other cellular components can be found in any standard biology textbook such as those by Karp [1] and Alberts et al. [2].

Table 1.1 Functional parallels between a cell and a manufacturing plant. Not all cellular components mentioned here are described in this chapter, but they are listed as a review and/or an impetus for further reading.

<i>Component of manufacturing plant</i>	<i>Analogous component(s) of cell</i>
Machines that make products	Ribosomes, enzymes
Doors	Pores, ion channels
Internal walls	Membranes
Central computer	DNA/chromosomes/genome
Central computer room	Nucleus (eukaryotic cells)
Combustion engine	Mitochondria (eukaryotic cells), cell membrane (prokaryotic cells)
Solar cell	Chloroplasts (eukaryotic cells), cell membrane (prokaryotic cells)
Packaging room	Golgi complex (eukaryotic cells)
Pipelines	Cytoskeleton and endoplasmic reticulum (eukaryotic cells)
Forklifts	Pumps, vesicles
Garbage disposal system	Lysosomes and vacuoles (eukaryotic cells)

1.2 Introduction to Information Handling in Cells

In order to reproduce themselves, organisms must transmit their design specifications faithfully to future generations. These specifications are their genetic information, contained within cells’ DNA. Therefore cells must store, retrieve, and copy this genetic information efficiently and precisely; in other words, they must act like tiny computers. Cells and computers address their information-handling problems in similar ways (Table 1.2); for instance, just as information on a computer is organized into discrete files, genetic information is divided into discrete units called genes. In general, one gene contains the instructions for making one polypeptide.

Table 1.2 Comparison of information handling by computers and cells. Adapted from the online tutorial, “Biological Information Handling: Essentials for Engineers” (www.biologyforengineers.org).

<i>Information-handling task</i>	<i>Computer solutions</i>	<i>Cellular solutions</i>
Storing source code	Computers store their source code as a binary code of zeros and ones.	Cells store their source code in DNA as a code of four nucleotide bases (A, C, G, and T).
Organizing source code	The source code involving specific outputs is organized in discrete segments, which are files.	The source code involving specific outputs is organized in discrete segments called genes.
Copying source code before use	A computer copies needed code into RAM (Random Access Memory) to speed up processing.	A cell copies parts of its DNA into an intermediate molecule, RNA, to speed up processing and minimize risk to the DNA.
Signaling where to begin copying	Computer code contains addresses for locating where to begin copying.	Cells use specific sequences of DNA, called promoters, to signal where to begin copying.
Pathway to generate output	Source code → temporary storage → output	DNA (source code) → RNA (temporary storage) → proteins (output)

(See [3] for a discussion of exceptions.) The cell's demand for a particular protein, each consisting of one or more polypeptides (see below), then determines how often the corresponding gene is accessed.

The handling of genetic information in cells can be thought of as a cycle in which the DNA code is transcribed to a similar molecule called RNA (ribonucleic acid), which is then translated to make proteins, which in turn are used to replicate, repair, and recombine the DNA. This information-handling cycle is known in biology as the Central Dogma (Figure 1.1), since it applies to all living organisms. (Retroviruses such as HIV have an RNA genome that replicates via a DNA intermediate; however, viruses are not considered organisms by most biologists.) More information about the individual stages of the cycle is offered below; this information is also available at the website www.biologyforengineers.org, which offers a free animated tutorial, "Biological Information Handling: Essentials for Engineers."

1.3 The Importance and Diversity of Proteins

A critical output of the Central Dogma is the production of proteins, which are molecular machines that carry out most of the cell's "work." Some proteins have a structural role; they are the bricks and mortar of a cell. Other proteins actively work to process nutrients and help the cell to grow, copy the cell's DNA, synthesize RNA, and direct cellular reproduction, among other tasks. Still other proteins have regulatory roles, serving as cellular switches that turn functions on and off.

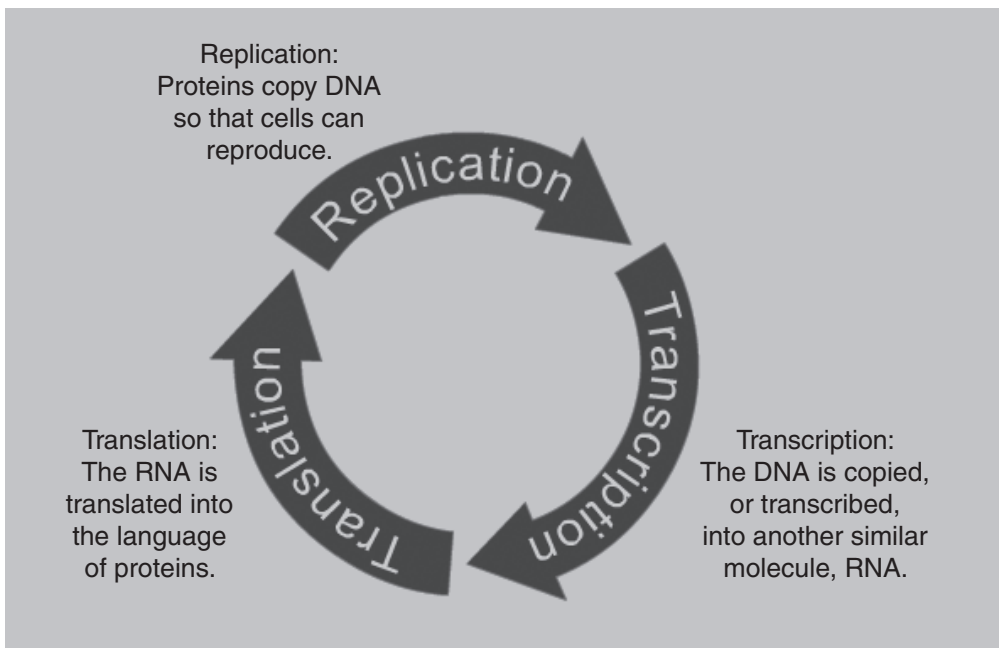


Figure 1.1 The Central Dogma of biology. DNA is copied into RNA (transcription); the RNA is used to make proteins (translation); and the proteins perform functions such as copying the DNA (replication). Image from the online tutorial, "Biological Information Handling: Essentials for Engineers" (www.biologyforengineers.org).

Many of the proteins in these different categories have the common feature of catalyzing a particular chemical reaction; these proteins are known as enzymes.

Proteins are polymers. Specifically, they are long chains of chemically diverse building blocks called amino acids. A single chain is called a polypeptide; a protein consists of one or more polypeptides that have folded into their functional three-dimensional shapes. The sequence of amino acids in each polypeptide chain (referred to as the primary structure) dictates the protein's final three-dimensional shape, although the rules by which this occurs are still not fully understood [4].

Proteins are an extremely versatile class of polymers, and the possible variations are almost limitless. For a protein that is 100 amino acids long, with 20 different amino acids possible at each position, there are 20^{100} (1.27×10^{130}) possible primary structures. Although the majority of these structures would not generate a functional polypeptide, the total number of combinations (total design space) is still so vast that the resulting proteins are capable of almost any function one can imagine. Depending on the organism, a cell may contain thousands to tens of thousands of types of proteins, each present in numerous copies [5].

1.4 DNA Replication: Copying the Code

Before a cell divides, it must copy its DNA so that its progeny will also be able to reproduce and function. This copying process is called replication.

So what exactly is DNA? DNA is a polymer of nucleotides; a nucleotide consists of a phosphate ($-\text{PO}_4^{3-}$) group, a five-carbon sugar (deoxyribose), and a nitrogen-containing base. Four types of these bases are found in DNA: adenine (A), cytosine (C), guanine (G), and thymine (T). A fifth base, uracil (U), is not present in DNA but is found in RNA (see below).

The three-dimensional structure of DNA consists of two strands of nucleotides spiraling around each other in a twisted-ladder structure usually described as a double helix [6]. The “rungs” of the ladder are the nitrogenous bases, whose chemical structures favor the pairing of A with T and C with G. This information is depicted schematically in Figure 1.2.

DNA replication is directed by an enzyme known as DNA polymerase. For replication to occur, the two strands of the double helix must come apart so that new strands can be synthesized alongside the existing strands, which function as templates. DNA polymerase then works its way along each template strand, attracting nucleotides complementary to those of the template strand and linking those nucleotides together to form a new strand (Figure 1.3). Once a given region of DNA is successfully copied, the old and new strands rewind into their familiar double helix shape; meanwhile DNA polymerase continues matching nucleotides to the template strands until the entire chromosome is copied. The cell now contains two identical sets of DNA, each made up of one new strand and one old strand. This replication pattern has been termed “semiconservative replication,” since one half of each double helix is retained (conserved) from the previous generation [7].

An interesting side note concerning DNA polymerase is that, although it is often described as moving along DNA strands like a train on a track, there is good evidence that it remains fixed in space while pulling the DNA past itself [8].



Figure 1.2 Schematic representation of DNA. Note the invariant pairing of bases: A is always complementary to T and C is always complementary to G. Image from the online tutorial, “Biological Information Handling: Essentials for Engineers” (www.biologyforengineers.org).

1.5 Transcription: Sending a Messenger

A cell’s DNA contains the instructions for making proteins. If the DNA were read directly by the protein-making machinery, however, the DNA could be damaged, and the process could be slow and difficult to regulate. To prevent these problems, the cell copies genetic information from its DNA into an intermediate called messenger RNA (mRNA) in a process called transcription. The mRNA then directs the synthesis of proteins via the process of translation.

Transcription is both similar to and distinct from DNA replication. The enzyme that carries out this process, RNA polymerase, acts like DNA polymerase in that it binds to an unwound section of DNA and synthesizes a new strand of nucleotides using the existing strand as a template. However, in the case of transcription, the newly created strand is mRNA, not DNA, and does not stay next to the DNA strand. Instead it heads off to a ribosome, the site of protein synthesis (see below). Also, while DNA polymerase copies a cell’s entire genome when the cell is ready to divide, RNA polymerase is much more selective in its copying; it only copies a particular gene when the corresponding protein is needed by the cell at that particular time. Sequences of nucleotides called promoters and terminators tell RNA polymerase where to start and where to stop copying, respectively.

The transcription process begins when RNA polymerase binds to a promoter region. It then attracts nucleotides complementary to those of the gene of interest, putting A’s across from T’s, C’s across from G’s, G’s across from C’s, and U’s (uracil, a base unique to RNA) across from A’s (Figure 1.4). RNA polymerase continues transcription until it reaches a terminator region, at which point the newly

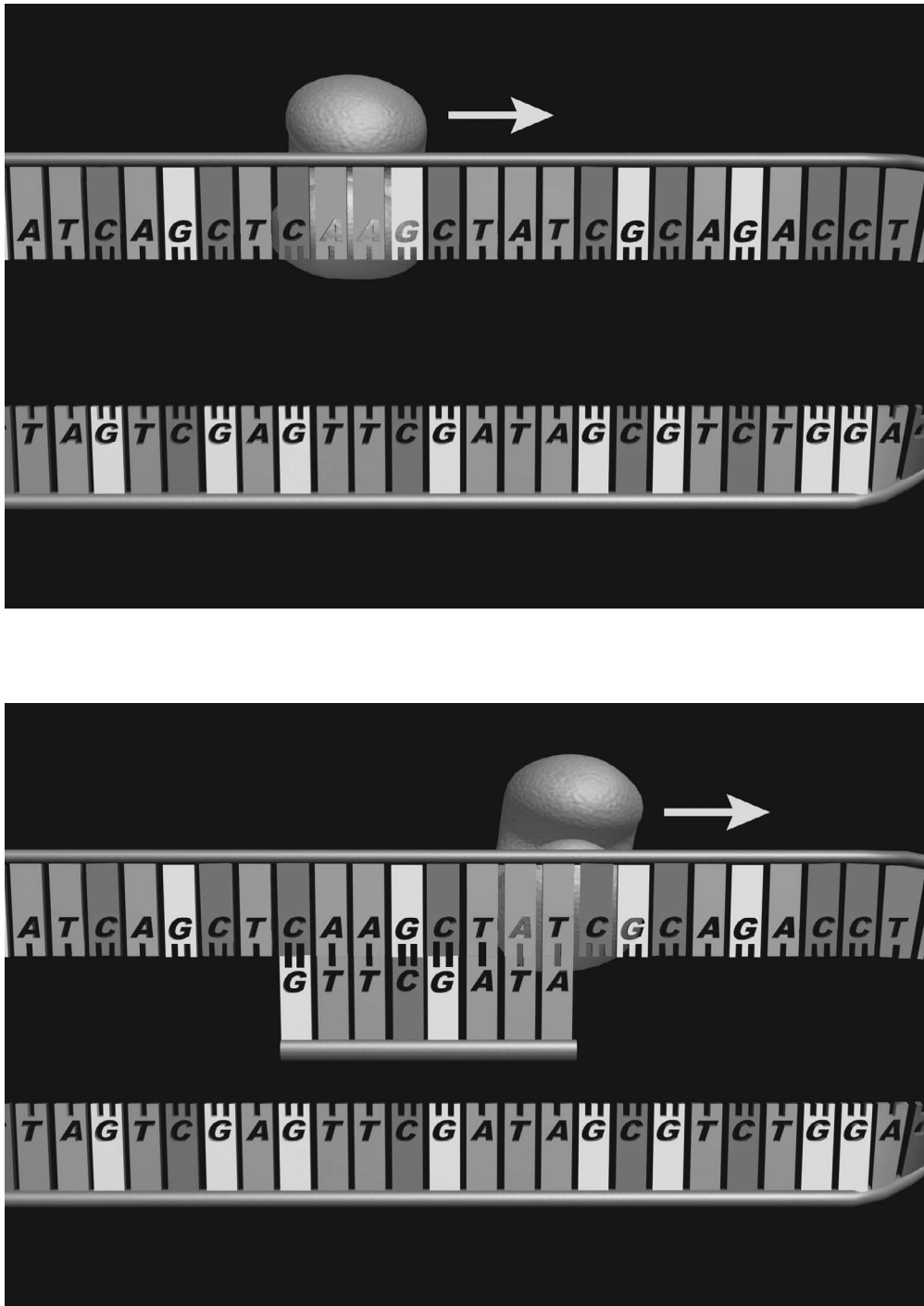


Figure 1.3 Replication of DNA by DNA polymerase. After the two strands of DNA separate (top), DNA polymerase uses nucleotides to synthesize a new strand complementary to the existing one (bottom). Images from the online tutorial, “Biological Information Handling: Essentials for Engineers” (www.biologyforengineers.org).

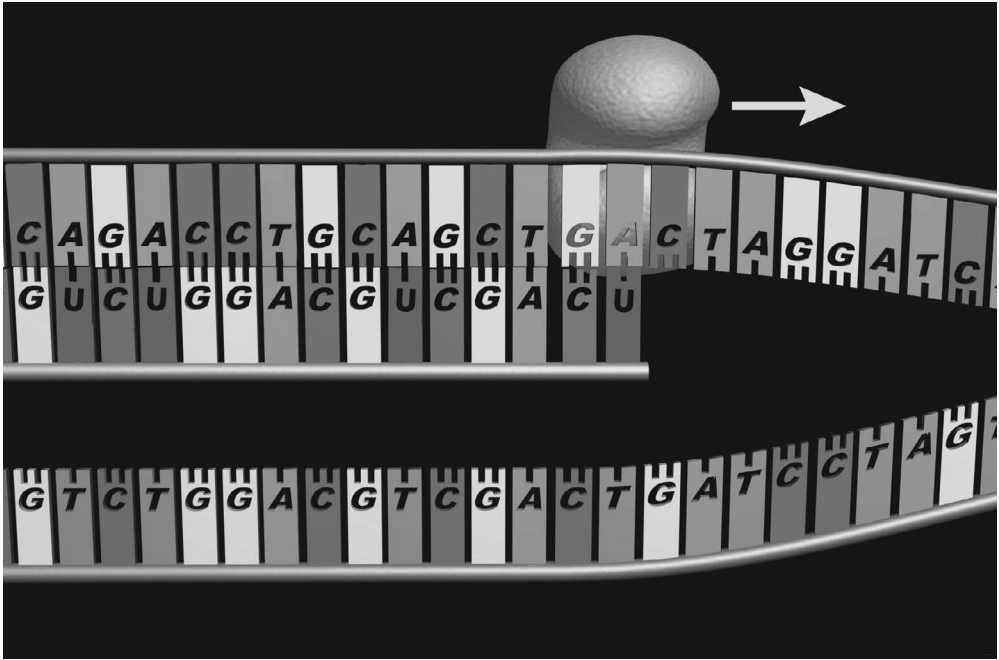


Figure 1.4 Transcription of DNA by RNA polymerase. Note that RNA contains U's instead of T's. Image from the online tutorial, "Biological Information Handling: Essentials for Engineers" (www.biologyforengineers.org).

made mRNA is released and RNA polymerase is free to find another gene in need of copying.

1.6 Translation: Protein Synthesis

In translation, the cell uses the genetic information contained in mRNA to make the proteins that carry out the cell's work. The cell translates the code contained in the mRNA into a new language, the language of proteins, which is based on amino acids. Two other types of RNA, ribosomal RNA (rRNA) and transfer RNA (tRNA), also assist in the protein-assembly process.

A cellular complex called a ribosome coordinates this process. A ribosome is made of both protein and RNA and consists of two parts, the large and small subunits, which clamp around the mRNA about to be translated. The ribosome brings together the mRNA and a set of adapter molecules called transfer RNAs (tRNAs), which carry the amino acids that will form a polypeptide chain. The tRNAs bring their amino acids to the mRNA in a specific order governed by the attraction between the mRNA codons—sequences of three nucleotides—and complementary nucleotide triplets on the tRNA called anticodons.

At the beginning of translation, the ribosome attaches to the mRNA (at a particular sequence called the ribosome-binding site) and then finds the mRNA's initiation codon, where translation starts. Since the sequence of this initiation codon is virtually always AUG, it attracts a tRNA with the complementary anticodon UAC (Figure 1.5). The tRNAs with this anticodon carry the amino acid methionine, so methionine will be the first amino acid in the polypeptide. In other words, the tRNA serves to “translate” the codon AUG into the amino acid methionine; AUG codes for methionine (Table 1.3).

Once this first tRNA is in place, the next mRNA codon becomes exposed, and a tRNA with the complementary anticodon binds to that codon. A peptide bond then forms between the amino acid bound to the first tRNA (methionine) and the amino acid bound to the second tRNA. At this point, the first tRNA dissociates from its amino acid, leaving the second tRNA holding the two-amino-acid chain. The process is then repeated for the third and subsequent mRNA codons. The ribosome advances along the mRNA, three nucleotides at a time, using a ratcheting mechanism; mRNA codons are matched up with tRNA anticodons; and each newly arriving tRNA brings an amino acid to add to the growing polypeptide chain.

Translation continues until the ribosome encounters a stop codon in the mRNA (Table 1.3). This nucleotide triplet signals that the polypeptide chain is complete. The stop codon causes all the components of translation to separate. The ribosome can disassemble and be used again. The mRNA is degraded back into its building blocks, the nucleotides. Meanwhile, the newly made polypeptide chain is further processed and folds into a mature, functional protein.

Although it is well established that mRNA nucleotides are translated in groups of three, it is interesting to consider the hypothetical alternatives. If each individual nucleotide—A, C, G, or U—coded for an amino acid, only four different amino

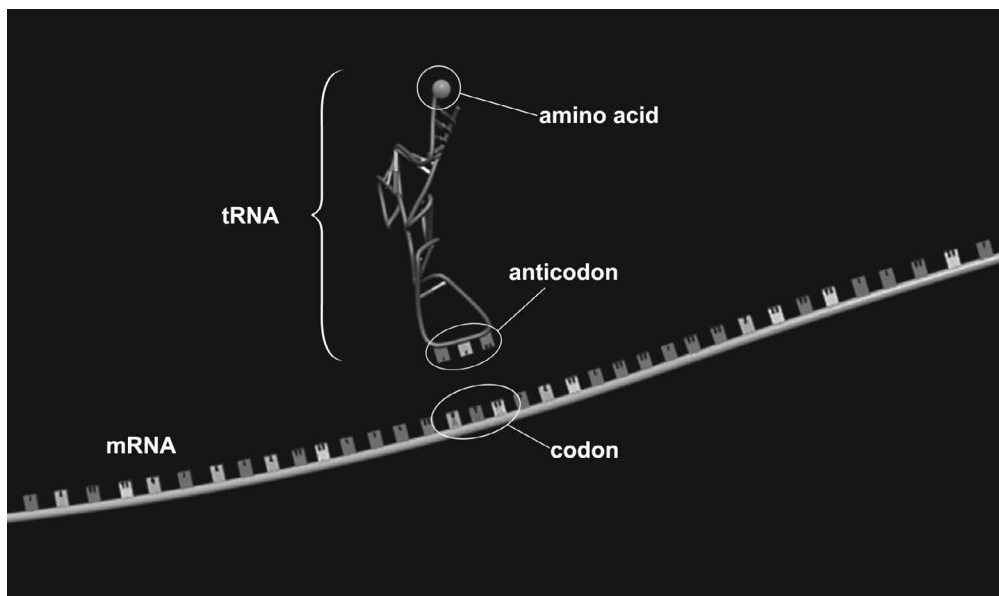


Figure 1.5 In the process of translation, each mRNA codon attracts a tRNA molecule containing a complementary anticodon. Image from the online tutorial, “Biological Information Handling: Essentials for Engineers” (www.biologyforengineers.org).

Table 1.3 The genetic code, as established by the pioneering work of Marshall Nirenberg, Robert Holley, and Har Gobind Khorana during the 1960s [9]. The 64 possible three-nucleotide codons in mRNA are translated into 20 different amino acids as shown below. For example, AUG codes for methionine and UGG codes for tryptophan. This translation process depends on the tRNAs, which link the codons to the amino acids via their anticodons.

<i>1st position</i>	<i>2nd position</i>				<i>3rd position</i>
	<i>U</i>	<i>C</i>	<i>A</i>	<i>G</i>	
U	Phenylalanine	Serine	Tyrosine	Cysteine	U
	Phenylalanine	Serine	Tyrosine	Cysteine	C
	Leucine	Serine	STOP	STOP	A
	Leucine	Serine	STOP	Tryptophan	G
C	Leucine	Proline	Histidine	Arginine	U
	Leucine	Proline	Histidine	Arginine	C
	Leucine	Proline	Glutamine	Arginine	A
	Leucine	Proline	Glutamine	Arginine	G
A	Isoleucine	Threonine	Asparagine	Serine	U
	Isoleucine	Threonine	Asparagine	Serine	C
	Isoleucine	Threonine	Lysine	Arginine	A
	Methionine	Threonine	Lysine	Arginine	G
G	Valine	Alanine	Aspartate	Glycine	U
	Valine	Alanine	Aspartate	Glycine	C
	Valine	Alanine	Glutamate	Glycine	A
	Valine	Alanine	Glutamate	Glycine	G

acids could then be used, which would limit protein diversity to far less than that noted above. If amino acids were specified by pairs of nucleotides such as AA or AU, a total of $4^2 = 16$ different nucleotide pairs, and thus 16 different amino acids, would be possible—still less than the 20 naturally occurring amino acids found in cells. Since the nucleotides are actually interpreted in groups of three, there are $4^3 = 64$ different possible codons, more than enough to cover all 20 amino acids. Thus some amino acids are coded for by more than one codon (Table 1.3).

1.7 Control of Gene Expression

The transcription and translation of genes into proteins is also known as gene expression. At any given time, a cell will only “express” the genes whose proteins are needed at the time, suggesting that transcription and/or translation are under tight control.

In theory, the concentration of a protein in a cell can be controlled by expediting or interfering with any of several processes: transcription of the gene by RNA polymerase, binding of ribosomes to the mRNA, degradation of the mRNA, degradation of the protein, and so forth. In general, though, altering RNA polymerase activity is the most important means of altering gene expression. This makes sense from an efficiency standpoint; if the cell does not need a protein, it’s simpler and less energetically demanding to simply stop transcribing the corresponding gene, rather than continuing to make the mRNA and then having to destroy it and any protein that is made from it.

There are two general classes of proteins that affect the activity of RNA polymerase: repressors and transcription factors. Repressor proteins bind to the promoter region of a gene and block RNA polymerase from transcribing it, thus reducing gene expression. Transcription factors have the opposite effect; they also bind to the DNA but *increase* transcription by RNA polymerase. The expression of any given gene can be controlled by one or more repressors or transcription factors or both.

The lactose metabolism enzymes in the bacterium *E. coli* offer a classic example of the control of gene expression [10]. Lactose is a sugar that *E. coli* can use as food; however, if no lactose is present in the environment, the lactose-processing enzymes are not needed, so the corresponding genes are not transcribed. In the absence of lactose, transcription is stopped by a repressor protein that binds to the promoter of these genes and restricts RNA polymerase's access to them. However, when lactose is present, a form of the lactose binds to the repressor and alters its shape so that it can no longer bind to the promoter. RNA polymerase can then transcribe the genes, leading to synthesis of the enzymes and digestion of the lactose.

This example of the lactose enzymes is a simplified one in that gene expression is portrayed as a digital (i.e., on-or-off switch) phenomenon. Some genes are indeed controlled in this binary manner, so it may soon be possible to build synthetic genetic circuits that use logic gates (AND, OR, etc.) to perform computations [11]. However, most genes are expressed in analog fashion, with many possible intermediate levels of expression, potentially complicating any such attempts at genetic computing.

1.8 Genetic Engineering

Since DNA replication, transcription, and translation are now well understood by biologists, they are relatively amenable to manipulation via genetic engineering. Perhaps the simplest form of genetic engineering is simply introducing a gene into a cell that did not previously have that gene. The foreign gene can then be transcribed and translated by the host cell's molecular machinery, leading to production of a foreign protein in the host. In this way, the bacterium *E. coli* has been used to produce large quantities of the human hormone insulin, which can then be harvested, purified, and given to diabetic people who cannot make their own insulin [12].

Another significant biotechnological advance was the development of the polymerase chain reaction (PCR) by Kary Mullis in the mid-1980s [13]. This test-tube technique induces DNA polymerase to create millions of copies of short stretches of DNA (usually 100 to 2000 nucleotides), thus enabling further analysis of this DNA. To permit rapid DNA copying at high temperatures, PCR employs a heat-stable DNA polymerase isolated from the heat-loving bacterium *Thermus aquaticus* [14].

While entire genes can be deleted from or added to cells with relative ease, much of today's genetic engineering entails combining parts of different genes in novel ways. In what is known as a transcriptional fusion, the promoter of one gene is fused to the coding region of another gene. This puts expression of the second gene under the control of factors that normally affect transcription of the first gene. Therefore, if you wish to understand the transcriptional control of a particular

gene, you can fuse that gene's promoter to the coding region of another gene whose protein product is easy to quantify—for example, because it catalyzes the formation of a colored chemical. You can then subject your cells to a variety of conditions and determine the extent of transcription (e.g., by measuring production of the colored chemical) under each condition [15].

Recent years have also brought about the increasingly common use of translational fusions, in which two normally independent polypeptides are combined into one protein. A translational fusion is created by splicing together the genes for each protein to create a new hybrid gene; this can then be transcribed and translated to yield a single long polypeptide consisting of the two original polypeptides joined together. In many cases, a protein of interest is fused to green fluorescent protein (GFP), whose location inside cells can easily be seen due to its fluorescence. GFP fusion proteins thus allow visualization of the movement and position of other proteins that could not normally be tracked [16].

Yet another frontier in genetic engineering is focused on novel RNA-targeted applications. Early efforts in this area have shown that translation of mRNA can be prevented with “antisense RNA” that binds to mRNA to which it is complementary, thus excluding it from ribosomes, and with catalytic RNA molecules known as ribozymes, which chop up mRNA before it can be translated. More recent research has uncovered additional mechanisms by which synthetic RNA can be used to either increase or decrease translation of specific genes [17].

1.9 Summary

A cell's DNA is a series of nucleotides containing the bases adenine (A), cytosine (C), guanine (G), and thymine (T). The nucleotide sequences of the genes in DNA contain instructions for making proteins, which are molecular machines that allow the cell to grow and reproduce. Proteins are made in two steps: transcription of DNA to form RNA, followed by translation of the RNA into polypeptides that fold into functional proteins. The incredible diversity of protein structures and functions reflect the almost limitless ways in which amino acids can be combined. Gene expression can be stimulated by transcription factors and prevented by repressors, ensuring that proteins are synthesized only when needed. Our considerable knowledge of information processing in cells has enabled rapid advances in genetic engineering, with further progress undoubtedly ahead.

Acknowledgments

The current chapter is based in part on the CD tutorial, “Biological Information Handling: Essentials for Engineers,” which was funded by a grant to M.E.L. from the HHMI Professors Program. Contributors to that CD included the authors of this chapter; David Farkas and Patricia Kirkham (University of Washington Department of Technical Communication), who edited it for clarity and organization; and Marc Hoffman and Bob Lindenmayer, who created the images and animations.

References

- [1] Karp, G., *Cell and Molecular Biology*, 4th ed., New York: John Wiley & Sons, Inc., 2004.
- [2] Alberts, B., et al., *Molecular Biology of the Cell*, 4th ed., New York: Garland Science, 2002.
- [3] Ast, G., "The Alternative Genome," *Sci. Am.*, Vol. 292, No. 4, 2005, pp. 40–47.
- [4] Contreras-Moreira, B., et al., "Empirical Limits for Template-Based Protein Structure Prediction: The CASP5 Example," *FEBS Lett.*, Vol. 579, No. 5, 2005, pp. 1203–1207.
- [5] Pandey, A., and M. Mann, "Proteomics to Study Genes and Genomes," *Nature*, Vol. 405, No. 6788, 2000, pp. 837–846.
- [6] Watson, J. D., and F. H. C. Crick, "Molecular Structure of Nucleic Acids," *Nature*, Vol. 171, 1953, pp. 737–738.
- [7] Meselson, M., and F. W. Stahl, "The Replication of DNA in *Escherichia coli*," *Proc. Nat. Acad. Sci. USA*, Vol. 44, 1958, pp. 671–682.
- [8] Lemon, K. P., and A. D. Grossman, "Localization of Bacterial DNA Polymerase: Evidence for a Factory Model of Replication," *Science*, Vol. 282, No. 5393, 1998, pp. 1516–1519.
- [9] Singer, M. F., "1968 Nobel Laureate in Medicine or Physiology," *Science*, Vol. 162, No. 852, 1968, pp. 433–436.
- [10] Jacob, F., "Genetics of the Bacterial Cell," in *Nobel Lectures, Physiology or Medicine 1963–1970*, Amsterdam: Elsevier, 1972.
- [11] Sprinzak, D., and M. B. Elowitz, "Reconstruction of Genetic Circuits," *Nature*, Vol. 438, No. 7067, 2005, pp. 443–448.
- [12] The, M. J., "Human Insulin: DNA Technology's First Drug," *Am. J. Hosp. Pharm.*, Vol. 46, No. 11, Suppl. 2, 1989, pp. S9–S11.
- [13] Mullis, K., et al., "Specific Enzymatic Amplification of DNA in Vitro: The Polymerase Chain Reaction," *Cold Spring Harbor Symp. Quant. Biol.*, Vol. 51, Pt. 1, 1986, pp. 263–273.
- [14] Brock, T. D., "The Value of Basic Research: Discovery of *Thermus aquaticus* and Other Extreme Thermophiles," *Genetics*, Vol. 146, No. 4, 1997, pp. 1207–1210.
- [15] Cui, C., et al., "Reporter Genes in Transgenic Mice," *Transgenic Res.*, Vol. 3, No. 3, 1994, pp. 182–194.
- [16] Gerdes, H. H., and C. Kaether, "Green Fluorescent Protein: Applications in Cell Biology," *FEBS Lett.*, Vol. 389, No. 1, 1996, pp. 44–47.
- [17] Isaacs, F. J., D. J. Dwyer, and J. J. Collins, "RNA Synthetic Biology," *Nat. Biotechnol.*, Vol. 24, No. 5, 2006, pp. 545–554.

Proteomics: From Genome to Proteome

Stephanie E. Mohr, Yanhui Hu,
and Joshua LaBaer

2.1 Defining the Proteome

2.1.1 From Genes to Proteins

The availability of whole-genome sequence has fundamentally changed the approach many researchers take in trying to understand the biochemical and biological functions of proteins. As whole-genome sequence is now available for hundreds of genomes, researchers can get a global picture of all of the proteins encoded by many genomes, including the human genome [1, 2]. The sequence of predicted proteins can then be used to identify conserved features such as subcellular localization signals, binding domains, and protein modification sites, and to predict protein function via comparison of predicted proteins to proteins of known function [3, 4]. Yet genomic information and sequence comparisons alone are not sufficient to provide a full understanding of how proteins function. Here, we outline and explore some important considerations that impact this process (Figure 2.1).

First, whereas the function of some proteins can be deduced by comparison to well-characterized proteins, the function(s) of many proteins are either only vaguely understood (e.g., predicted to have some enzymatic function, but its target(s) and regulation are unknown) or completely unknown as their sequences are novel. Second, even when function can be predicted or inferred, proteins may have additional and/or cell-specific functions that cannot be deduced by sequence analysis alone. Third, any given tissue or cell is likely to express only a subset of the genes encoded by the genome and, moreover, may express different splice variants, thus complicating efforts to uncover the individual and cooperative functions of proteins in a particular cell type (Figure 2.1). A further complication is that the relative abundance of a protein product of a gene may be different in one cell type compared with another due to cell-specific regulation of transcription (the process by which genes are expressed to produce the mRNA templates used for protein production), translation (the process by which the mRNA templates are used to make proteins), and/or protein degradation. Although DNA microarrays (which measure mRNA

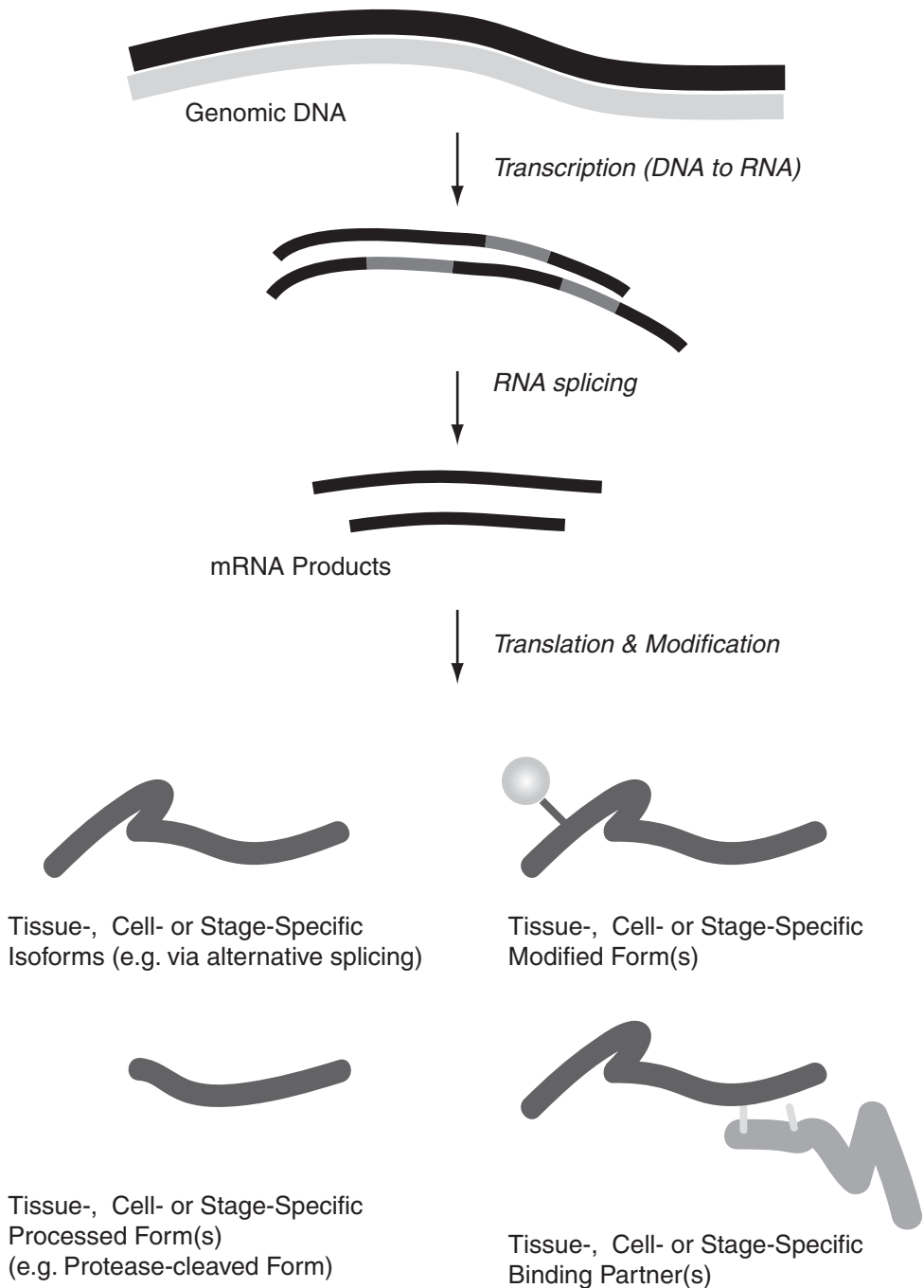


Figure 2.1 Complexity at the protein level exceeds complexity at the gene and transcript levels. Individual genes in the genome are transcribed into RNA. In eukaryotes, the RNA may be further processed to remove intervening sequences (RNA splicing) and results in a mature transcript that encodes a protein. Different proteins may be encoded by differently spliced transcripts (alternative splicing products). Moreover, once proteins are produced, they can be processed (e.g., cleaved by a protein-cutting protease) or modified (e.g., by addition of a sugar or lipid molecule). In addition, proteins may have noncovalent interaction with other proteins (and/or with other biomolecules such as lipids or nucleotides). Each of these can have tissue-, stage-, and cell-type specific effects on the abundance, function, and/or stability of proteins produced from a single gene.

template levels for thousands of genes simultaneously) are helping researchers to compare mRNA expression levels, these levels do not often correlate with protein levels [5, 6]. Moreover, some arrays fail to detect functionally relevant alternative mRNA forms, which result from splicing different elements of the gene sequence together in alternate combinations. Thus, figuring out which proteins exist in which forms at what levels in which cells remains a challenge.

Proteins may also exist in different functional states in different cellular contexts. Changes made to the protein after its synthesis, called post-translational modifications, include protease cleavage or the addition of phosphate, sulfate, lipid, or sugar groups. These modifications can have dramatic effects on function, localization, enzymatic activity, and/or stability and can be tightly regulated in stage-, tissue-, and cell-type specific patterns (Figure 2.1). Similarly, noncovalent protein-protein, -DNA, -RNA, or -lipid interactions also affect function, localization, activity, and/or stability and can be tightly regulated (Figure 2.1). Thus it is clear that the level of complexity increases dramatically as one moves from looking at the genome (the collection of all genes) and the transcriptome (the collection of all mRNA transcripts) to looking at the “proteome,” or the complete set of proteins expressed by a given cell or organism (Figure 2.1). Taken together with the desire to understand protein function not only in normal cells but also in specific disease states, the task of understanding protein function becomes very large indeed. The urgency to identify and characterize proteins is highlighted by the fact that the vast majority of successful therapeutic drugs are directed against proteins, rather than against genes or transcripts [7–9].

2.1.2 What Is Proteomics?

Stated briefly, “proteomics” is the large-scale study of proteins. Traditionally, proteomics research has focused on identifying and quantifying proteins by isolation from cell extracts (e.g., via two-dimensional gel electrophoresis or liquid chromatography) followed by protein identification (e.g., via mass spectrometry and the related MALDI and electrospray approaches) [7]. However, some proteins cannot easily be studied using the current methods for separation and identification [10, 11], and both the sheer number of different proteins in a cell (which in eukaryotic cells is thought to be in the range of 10,000 different proteins) and the dynamic range of protein levels (which can differ by up to a millionfold) limit the range of proteins that can be detected using these methods [7]. Moreover, the focus of this approach yields data primarily on protein abundance, without necessarily elucidating the biological role of the proteins. Fortunately, genomic research has alleviated some of the need to identify proteins *en masse* and enabled new investigation into functional analyses.

The scope of proteomics has expanded in recent years and now includes both the traditional, “abundance-based” approaches and “function-based” approaches [12]. Proteomics research now overlaps many other fields of biology in terms of its exploration of protein structure, protein interactions, signal transduction, enzymatic activity, among others, and finds common ground with classical genetics in the sub-field of “functional proteomics” that is the focus of this chapter. In addition, proteomics is increasingly reliant on robotics and computer engineering, which make

it possible to carry out, track, and interpret large-scale studies. Finally, proteomics forges new ground in the scale and scope of the approaches encompassed by the term, and many proteomics researchers are part of an effort to integrate diverse data types toward a more general understanding of biology [13].

2.1.3 Functional Proteomics

Functional proteomics uses high-throughput, large-scale approaches to learn about protein function. The study of proteins long predates our knowledge of DNA and its structure, and biologists have accumulated numerous types of *in vitro* and *in vivo* functional approaches to study (1) protein structure; (2) protein-protein, protein-DNA, protein-RNA, and protein-lipid interactions; (3) drug interactions; (4) enzymatic activity and enzyme-substrate interactions; and (5) antigenicity (the ability of a substrate to produce an immune response in an organism). These methods were initially developed to study proteins at the individual protein level, whereas functional proteomics seeks to scale up these studies so that thousands of proteins are handled simultaneously, often using automation and informatics. Of course, this creates interesting challenges for engineers and informaticians, who must manage data handling, data integration, resource distribution, and data analysis at this ‘high-throughput’ scale.

An axiom of studying protein function is that to study proteins, one must first isolate and/or produce them. In the modern era, protein production is accomplished by producing “recombinant” protein from a cloned copy of the gene that encodes a particular protein. Typically these cloned gene copies reside in plasmid vectors, which are circular DNA molecules that harbor the gene and enable its selective propagation and amplification in a cellular host such as bacteria. It follows, then, that an important prerequisite to the high-throughput approach of functional proteomics described above is the availability of large collections of cloned genes that can be used for the *in vitro* and *in vivo* production of many proteins [14, 15]. Below, we describe the construction, validation, and analysis of these gene collections, and provide examples of how these clones can be used in a wide array of functional proteomics approaches.

2.2 Building Gene Collections for Functional Proteomics Approaches

As stated above, a number of proteomics approaches require the availability of large sets of protein-coding genes as a first step in carrying out protein production in cell-free or cell-based systems [14, 16]. However, availability and use of these resources is limited, at least in part because they are challenging to produce, maintain, and distribute. Several large-scale gene collections have been attempted with varying success, including mammalian gene clone sets (reviewed in [14]) and clone sets for other organisms [17–24], which have been used successfully in a number of functional proteomics approaches. Nevertheless, there remains a clear need to produce additional, high-quality clone collections to facilitate study of diverse aspects of biology and biomedicine [15].

A flow chart of the basic steps involved in creating a gene collection are outlined in Figure 2.2 and a glossary of terms is provided in Table 2.1. Briefly, to produce clone collections, researchers must decide which genes to clone; that is, identify a target gene set, select the best representative DNA sequences for the genes they wish to capture, amplify each gene from a DNA template and capture it in an appropriate plasmid vector for propagation, and verify that the cloned genes will accurately encode full-length proteins without mutations. To produce the proteins experimentally, the genes must often be transferred from their first cloning vector into a specialized vector that is optimized for protein production in a specific application. During the cloning process, aberrant nucleotide changes can corrupt the gene sequence. It is essential to detect these changes and to determine if they will

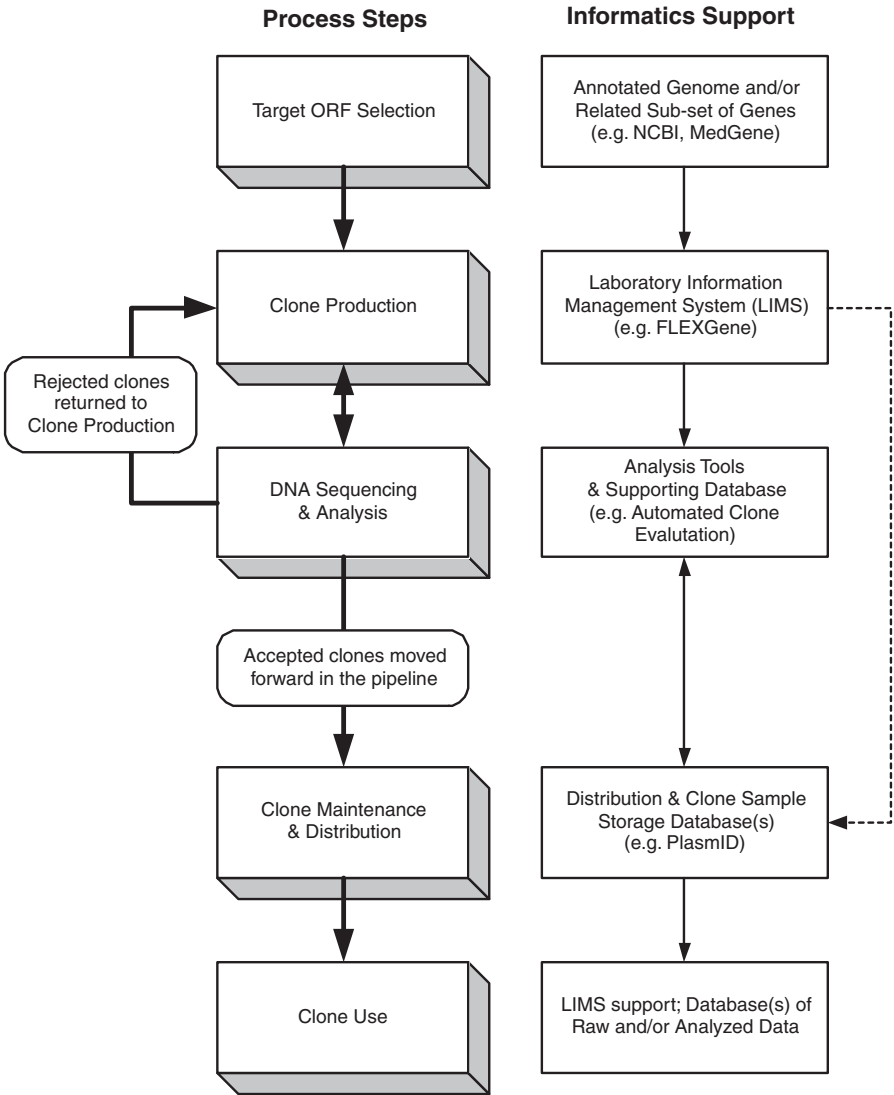


Figure 2.2 Flowchart of the major steps involved in gene clone set production and use.

Table 2.1 Common molecular biology terms used in this chapter.

<i>Term</i>	<i>Definition</i>
Bacterial colony (or clonal isolate)	Colony or “dot” of bacteria on a solid growth medium (e.g., agar dish) originating from a single bacterium.
Bacterial culture (or liquid culture)	Growth of bacteria in liquid medium. Bacteria are provided a defined nutrient broth and maintained at an optimal temperature, sometimes with agitation. When a plasmid vector is present and contains an antibiotic resistance gene (this is common), the appropriate antibiotic is added to the broth. This ensures that the bacteria maintain and propagate the plasmid.
Bacterial transformation	Introduction of a DNA fragment into bacterial cells, such as introduction of a circular plasmid, which will then be maintained and propagated if grown under the proper conditions (see bacterial culture).
Clone and subclone	In the context of this chapter, “clones” are unique isolates of a specific gene, often captured in a plasmid vector. Each clone can be used to make more copies of itself. Transfer of a clone (e.g., a specific gene) into a separate vector creates “subclones,” which are also identical in the subcloned region.
DNA purification	In the context of this chapter, “DNA purification” refers to a multistep process, often involving liquid handling and filtration steps, that is used to extract and isolate plasmid DNA from a bacterial culture in advance of sequencing, subcloning, or other steps.
Gel electrophoresis	Method for separating DNA, RNA, or protein using an electrical charge to separate DNA molecules through a three-dimensional matrix. The larger the DNA fragment, the slower it will move through the matrix. DNA isolated on a gel can be recovered and purified away from the matrix.
Glycerol stock	Liquid bacterial culture with glycerol added in order to facilitate long-term viability when stored frozen at -80°C (recovery by inoculation into liquid medium or spreading on agar growth media in a dish).
Oligonucleotide primer	Short DNA fragment [typically 18 to 30 base pairs (bp)] designed to anneal to a specific sequence. Pairs of primers are used for PCR (see below); individual primers are used to initiate DNA sequencing.
Plasmid vector (or vector)	Circular DNA fragment that has all of the necessary features for maintenance and propagation in bacteria and that typically includes a positive selection marker (e.g., an antibiotic resistance gene; see bacterial culture).
Polymerase Chain Reaction (PCR)	A method for amplification of a specific DNA fragment in which paired DNA strands are separated (by high temperature) and then each is used as a template for production of a complementary strand by an enzyme (a DNA polymerase). The specific sequence amplified is defined by the oligonucleotide primers included in the reaction mix, which hybridize to specific locations on a DNA fragment. Cycles of separation and enzyme-mediated polymerization are used to make many copies of the original template sequence.
Recombinational cloning	Strategy for moving a DNA fragment into a plasmid vector via a sequence-specific, enzyme-mediated event that can be carried out via a small volume in vitro reaction mix.
Sequence contig (or contig)	DNA sequence compiled from alignment of multiple, overlapping DNA sequence traces
Sequence trace (or trace file)	Direct readout (raw data) from an instrument that determines the order of base-pairs on a DNA fragment. In the readout, each base pair (A, C, G, and T) is assigned a specific color, and the trace indicates which color is most dominant at each position along the DNA fragment. The more dominant the color at a given position (i.e., highest peak), the more confident one can be that the correct base pair has been identified for that position along the fragment.

affect the final protein such as by deleting some amino acids (deletions), adding extra amino acids (insertions), prematurely truncating the protein, or introducing amino acid substitutions that alter protein function. Because they operate at large scale and demand a high standard for sequence integrity, the most successful cloning projects rely heavily on automation and informatics.

2.2.1 Selection of Target Genes for a Cloning Project

With more than 300 whole genomes sequenced and tools for gene annotation much improved, there is now a wealth of information about protein-coding genes. Scientific interest will typically drive the key early decision of which target organism to select and, furthermore, which set or subset of protein-coding genes should be represented in the collection. In the case of bacteria, it is feasible to plan and execute genome-scale cloning projects. For eukaryotes, however, the current operational scale and financial limits on cloning projects usually require selection of a subgroup of target genes for the cloning project. Approaches to target selection include use of experimental, computational, and/or data mining-based approaches to defining a set of genes of interest, as outlined in Figure 2.3.

2.2.1.1 Target Genes from an Annotated Bacterial Genome Sequence

Compared to eukaryotes, the genomes and proteomes of bacteria are relatively small. In addition, mRNA splicing does not occur in bacteria, and thus each gene lies as a contiguous string of characters in the genomic DNA. Therefore, amplifying genes directly from bacterial genomic DNA, which is easy to obtain, will yield an uninterrupted protein-coding sequence. Moreover, combined with the advantage that all genes are represented equally on this template, genomic DNA is an ideal template for producing bacterial clone sets. A critical step in designing the cloning strategy

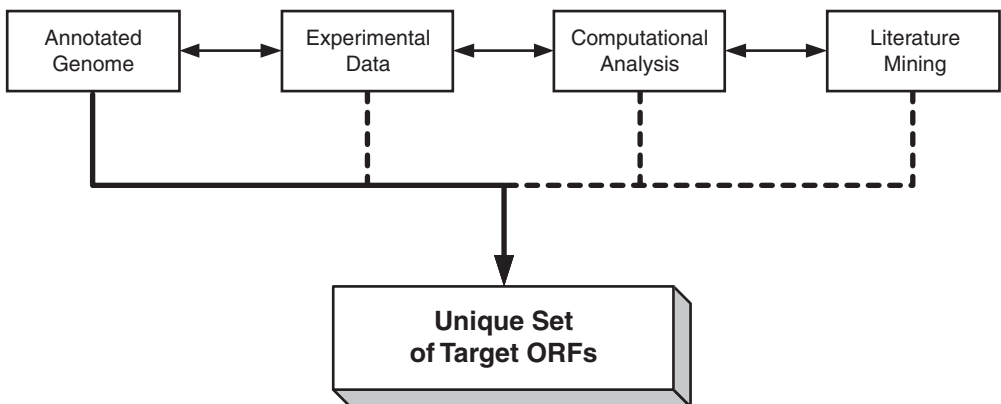


Figure 2.3 Bioinformatic approaches to select target genes for a cloning project. For bacterial genomes, target selection is drawn primarily from genome sequence, where introns are not a consideration and genome-scale projects are feasible. For eukaryotes, researchers commonly use one or more informatics-based methods to identify subgroups of target genes that share a common feature, such as function, localization, expression, or disease association. As noted, these information sources draw significantly on one another (as experimental data is in genome annotation).

is to ensure access to the most complete and accurate genome annotation available for the selected bacterium because amplification of gene sequences requires an exact match between the expected gene sequence (used to design oligonucleotide primers included in the amplification reaction) and the actual genomic sequence (to which the primers must anneal).

As many genes have been identified computationally but not yet experimentally validated, annotations can change over time as additional experimental data becomes available and is used to update gene predictions [25]. The National Center for Biotechnology Information (NCBI; <http://www.ncbi.nlm.nih.gov/Genomes>) lists in-process or complete genome sequences and gene annotations for more than 1,000 bacterial species. Additional information is available at the Comprehensive Microbial Resource website of The Institute for Genome Research (TIGR; <http://cmr.tigr.org/tigr-scripts/CMR/CmrHomePage.cgi>). Many well-studied microorganisms have organism-specific databases built from consortium efforts that benefit from dedicated input from their research communities. An example among bacteria is the *Pseudomonas* Genome Project (PGP; <http://www.pseudomonas.com>), which includes information about the opportunistic human pathogen *Pseudomonas aeruginosa* [26]. Indeed, the latter database was used as a source of information for compiling a set of 5,570 genes for a genome-scale *P. aeruginosa* protein-coding clone project [17].

2.2.1.2 Target Genes from Curated Subgroups of Eukaryotic Genes

The genomes of eukaryotic organisms such as *Drosophila melanogaster*, *C. elegans*, and humans are much larger than bacterial genomes, with 13,600, 19,000, and 22,000 predicted genes, respectively [2, 27–29], and the genes themselves are often significantly larger as well. Moreover, the protein-coding sequences of most eukaryotes are not contiguous on the genome but are disrupted by intervening introns, which are removed via RNA splicing to generate the mature mRNA transcript. Thus, genomic DNA cannot be used as a template for amplification of the protein-coding sequences. Instead, researchers must clone genes from cDNA, which is produced by converting mRNA (which has been spliced to create a contiguous protein-coding sequence) back into DNA. The mRNA is typically isolated from one or more cell types and represents a molecular snapshot of which genes are abundantly (or rarely) expressed in those cells. This raises the additional problem that copy numbers of various genes can vary dramatically in the cDNA. Researchers interested in low-abundance genes must sift through many copies of abundant genes to find them. When selecting a target gene set for a cloning project for eukaryotes, relative abundance and/or the availability of a previously cloned and isolated cDNA template is an important consideration.

2.2.1.3 Several Information Sources Can Be Used to Choose a Target Gene Set

Not all cloning projects strive for complete genome coverage, however, and choosing an appropriate subset of genes for a cloning project will be guided largely by scientific interests. Selections are often based upon one or more of the following:

(1) experimental data, such as a set of genes shown to be up-regulated or down-regulated via DNA microarray; (2) computational prediction from gene or protein sequence, such as identification of proteins that share a common functional domain; and (3) mining of published reports to identify proteins that share some feature such as structure, function, localization, pattern of expression, or disease association (Figure 2.3). The use of experimental data sets is fairly straightforward. Any data set in the lab can be selected, the reference sequences can be drawn from publicly available databases (see Section 2.2.1.6), and if appropriate, the list can then be cross-referenced with available cDNA templates. From a bioinformatics perspective, the other two approaches described above are perhaps the more interesting cases.

Computational predictions of function include comparison of primary protein sequence with a set of proteins of known function or a set of conserved motifs or domains [30] and computation of properties such as hydrophobicity based on amino acid composition of the primary protein sequence. Several freely available, on-line software tools and databases facilitate comparative and computational analyses and thus serve as a resource for identifying functionally related groups (Table 2.2). An instructive example is identification of the complete set of transmembrane (TM) proteins encoded by the human genome. Membrane-bound receptors and channel proteins have proved fruitful targets for therapeutic drugs, and there are many avenues of research for which a set of TM protein-producing clones

Table 2.2 Commonly used databases of information related to protein function, modification, and/or structure.

<i>Database</i>	<i>Content</i>	<i>URL</i>
Pfam	Protein domain and families	http://pfam.wustl.edu/
PRODOM	Protein domains	http://protein.toulouse.inra.fr/prodom/current/html/home.php
SMART	Protein domain and motifs	http://smart.embl-heidelberg.de/
PROSITE	Functional motifs	http://ca.expasy.org/prosite/
SignalP	Prediction of signal peptide cleavage sites	http://www.cbs.dtu.dk/services/SignalP/
Predotar	Prediction of mitochondrial and plastid targeting sequences	http://urgi.infobiogen.fr/predotar/
MITOPROT	Prediction of mitochondrial targeting sequences	http://ihg.gsf.de/ihg/mitoprot.html
TMHMM	Prediction of transmembrane helices	http://www.cbs.dtu.dk/services/TMHMM/
SOSUI	Prediction of transmembrane helices	http://sosui.proteome.bio.tuat.ac.jp/sosuiframe0.html
PSORT	Prediction of subcellular localization	http://www.psort.org/
TRANSFAC	Transcription factors and their genomic binding sites	http://www.gene-regulation.com/
DBD	Transcription factor prediction based on conserved domains	http://dbd.mrc-lmb.cam.ac.uk/skk/Cell2/index.cgi?Home

might be useful. Membrane-spanning domains have distinct patterns of hydrophobic and polar amino acids; thus, TM proteins can be identified using computational prediction of hydrophobicity and other properties of individual amino acids. Commonly used tools for TM domain prediction include TMHMM (for TM Hidden Markov Model) and SOSUI [31, 32]. Using these software tools and/or other defined criteria, a list of putative TM proteins can be generated and used as a set of target genes.

2.2.1.4 Mining Published Reports in Free Text Format

Mining published reports is another way to identify genes that share common attributes. Knowledge of biological systems exists in the form of millions of published literature citations in free text format. However, these data are difficult to handle in a high-throughput manner. One reason is that the data set is incredibly large (citations go back at least four decades for many research journals); another is the inherent redundancy and ambiguity of language that complicate text-based searches (for example, “was” is both the symbol for the human Wiskott-Aldrich Syndrome gene and a ubiquitous English-language word). One solution is systematic curation by well-trained experts who “translate” published information into a controlled vocabulary such as Gene Ontology (GO) terms [33]. The large-scale GO curation project has yielded a commonly used resource for identification of proteins with particular functions or subcellular localizations. An example is the set of human kinases, which add phosphate groups to one or more protein, lipid, nucleotide, or sugar substrates. Mining of the GO database, which in this case can be supplemented by mining of a conserved domain database, is sufficient to identify a set of approximately 700 putative human kinases, an approach that was used by one group to create a human kinase protein-coding gene collection that has already proved valuable for research [34].

2.2.1.5 Automated Mining of the Published Literature

Another solution to the problem of navigating the published literature is to develop text-mining tools that recognize specific sentence patterns in order to extract information automatically. Methods for extracting biological information from the scientific literature include extracting information about protein-protein interaction networks [35, 36]; linking genes to phenotypes [37]; linking genes to specific diseases [38, 39]; summarizing transcript diversity [40]; assigning protein functions, such as phosphorylation [41, 42], and building biological pathways [43]. The MedGene tool, for example, summarizes and organizes the biomedical literature and can be used to identify genes or proteins relevant to specific diseases (<http://hipseq.med.harvard.edu/MEDGENE/login.jsp>; [39, 44, 45]). This tool has been used to identify a set of more than 1,000 candidate genes linked to breast cancer (the “BC1000”), for example, and the resultant target set was successfully cloned and used in a study of oncogenesis [18]. Finally, the different methods for identifying related sets of genes are naturally interlinked and not exclusive—such that compilation of an appropriate target gene can involve input from genomic, experimental, computational, and/or

data-mining approaches so that the best possible predictive measures are used and compared before the work of clone prediction begins (Figure 2.3).

2.2.1.6 Assembling Reference Sequences for a Cloning Project

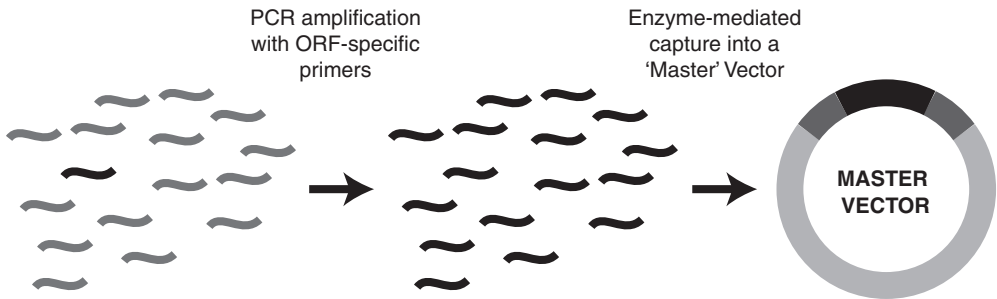
Once target genes have been selected, the next step is to download the relevant set of reference sequences, which will be used both to design the cloning strategy (i.e., in the design of oligonucleotides for gene amplification; see Section 2.2.2.1) and to define the standard sequences to which the experimental sequences obtained in the cloning project will be compared (see Section 2.2.3). Depending on the researchers' intentions, it may be important to avoid redundancy (there are often many sequence entries for the same gene), to ensure full length sequence (many entries are for partial genes), and to ensure accuracy (as sequencing projects vary with respect to completeness and quality). GenBank is the primary source for annotated gene sequences and includes DNA sequences submitted by both individual labs and large-scale sequencing efforts. Data are exchanged between GenBank, EMBL Data Library, and the DNA Data Bank of Japan to achieve comprehensive worldwide coverage, resulting in considerable redundancy. The NCBI Reference Sequence (RefSeq) database, by contrast, contains only curated sequences derived from GenBank records.

RefSeq is usually considered to be the best source for template sequences relevant to organisms represented in the RefSeq database, such as template sequences relevant to human, mouse, and rat genes. Moreover, when existing individualized cDNA templates can be used to clone genes, their template sequences can be compared to RefSeq to help determine if they are full length and correct. In addition, carefully curated genome annotations are available for many of the most-studied organisms (for examples, see [26, 46–49]). NCBI and TIGR both maintain comprehensive annotated bacterial genome sequences (see Section 2.2.1.1). In all cases, informaticians must develop tools to download and parse the data from these databases into formats compatible with their own local database(s), which will house the reference sequence data. Regularly scheduled checks for updates to gene annotations may also be appropriate.

2.2.2 Clone Production

Three major challenges confront high-throughput clone production: achieving a chemistry accurate enough to maintain the high standard needed for sequence integrity; developing a capture method and plasmid vector system robust enough to work reproducibly above 95%; and managing the scale of operation, which for large-scale efforts can include thousands of reference sequences, tens of thousands of unique clones, and hundreds of thousands of sequencing reads. In addition, during the design phase it may be valuable to consider using a scheme that facilitates enzyme-mediated transfer of gene inserts, so that it is easy to move the gene from a master vector in which it was captured into any number of different expression vectors (Figure 2.4). The enzyme-mediated transfer process is both automatable and error free, obviating the need to resequence any daughter clones once the parent

A. Amplification of ORFs from template DNA



B. Transfer of ORFs from master to expression vectors

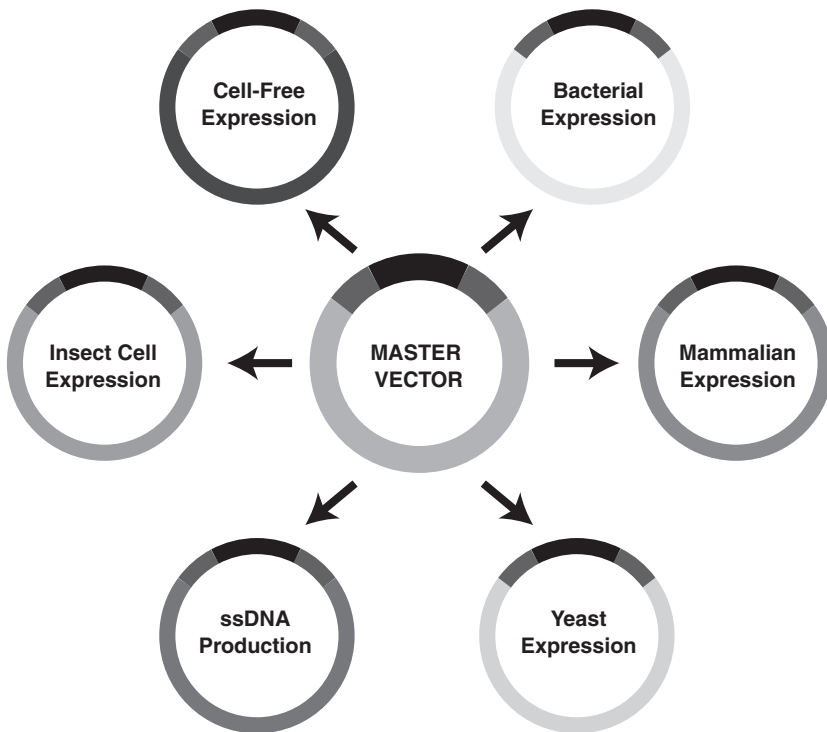


Figure 2.4 Cloning genes into plasmid vectors for propagation and use. (a) The target sequence of interest is amplified using gene-specific oligonucleotide primers and the DNA fragments are then captured in a plasmid vector that facilitates propagation of the gene sequence in bacteria. (b) Once a "master" clone has been completed, it can be used to generate any number of "expression" clones that make it possible to produce the protein encoded by the gene in specific cell-free or cell-based systems.

clones have been verified. At least two systems for this kind of site-specific, enzyme-mediated transfer are commercially available, and researchers have reported high-throughput transfer of inserts from the donor vector to expression vectors with as much as 100% success [19].

2.2.2.1 Gene Amplification, Capture, Isolation, and Sequencing

Although other approaches have been used with varying success (reviewed in [50]), most cloning efforts now use a PCR-based approach in which protein-coding sequences are amplified from DNA templates with oligonucleotide primers, which are targeted specifically at the gene of interest and which selectively amplify only the relevant protein-coding sequences. The resultant DNA fragments are captured in plasmid vectors, which facilitate propagation of DNA sequences in bacteria and facilitate a wide variety of experimental approaches (Figure 2.4). For large-scale clone production, production steps must be tracked using a robust laboratory information management system (LIMS or LIM System), which typically uses barcoding of containers and an underlying database to track the real-time whereabouts and progress of every clone in the system. This in turn is most readily accomplished when using automation to manage all steps in the process pipeline, which dramatically reduces errors and generates a log file record. In a perfect world, full line automation would be used, but most labs still use workstation automation, which has the added overhead of requiring users to log their process containers in and out of the LIMS at each step. The steps in the clone production pipeline used at the Harvard Institute of Proteomics (HIP) are outlined in Table 2.3 and can serve as a model of cloning projects that include single-colony isolation and sequence verification.

2.2.2.2 Automation for Clone Production

High-throughput clone production relies on a number of specialized automated systems to handle plates at the PCR, liquid culture, single-colony selection, and other steps (Table 2.3; Figure 2.5). Most of the steps involved in clone production are carried out in 96-well microtiter plates (eight rows by twelve columns; Figure 2.5). These standardized plates conform to a uniform footprint so that most liquid-handling robotics equipment can be used to handle all or most plate types. However, despite the standard, the plates vary from manufacturer to manufacturer. In

Table 2.3 A clone production pipeline.

<i>Production Step</i>	<i>Required</i>	<i>Automation</i>	<i>Informatics Support</i>
1 Design PCR primers (use reference sequence to design matching primers to amplify the gene)	Yes	not applicable	Nearest neighbor algorithm to calculate melting temperature; Determine plate and well positions; Generate barcode labels; Text output for primer ordering from vendor
2 Receive primers from vendor	Yes	Barcode scan	Laboratory Information Management (LIM) System update (primers received)

Table 2.3 (*continued*)

<i>Production Step</i>	<i>Required</i>	<i>Automation</i>	<i>Informatics Support</i>
3 Dilute primers (condition needed for cloning reactions)	Yes	Liquid handling robot in simultaneous 96-well format	Generate barcode labels; Select dilution protocol; LIM System update (primers diluted)
4 Prepare templates (arrange matching templates to facilitate 96-well transfers)	Yes ¹	Liquid handling robot with individualized addressable tips (rearray) ¹	Generate rearray file matching templates to primers; ¹ Generate barcode labels; Generate work list; System update (templates prepared)
5 Initiate first PCR amplification step ²	Yes	Liquid handling multi-reagent addition; Thermocycling (PCR)	Generate barcode labels; Select PCR protocol; System update (PCR 1 run)
6 Initiate second PCR amplification step ³	No	Liquid handling reagent addition; Thermocycling (PCR)	Generate barcode labels; Select PCR protocol; System update (PCR 2 run)
7 Gel isolate PCR products (assess amplification success—purify product if needed)	Yes	Electrophoresis system compatible with 96-well plate format; Robotic gel loading; Digital image capture	Capture PCR results into database (annotations on product size); Capture digital image files; System update (sample-based success/failure)
8 Gel purify PCR products (extract amplified DNA from gel matrix)	No	Plate-compatible centrifuge	Generate barcode labels; System update (step completed)
9 Capture PCR products into plasmid vector	Yes	Liquid handling reagent additions, temperature-controlled incubation	Generate barcode labels; System update (step completed)
10 Transform bacteria (introduce completed gene clone into bacteria)	Yes	Liquid handling reagent addition and dispense onto specialized 48-sector incubation agar dishes	Generate barcode labels; System update (step completed)
11 Isolate colonies (select one or more individual bacterial colonies from agar dish)	Yes	Automated digital imaging and analysis for colony detection; Robotic colony selection and transfer to liquid culture (includes barcode read, colony count); Liquid handling	Generate barcode labels; Capture colony count (measure of successful transformation); System update (colonies obtained)
12 Grow liquid culture (growth of bacteria containing new plasmids in 96-well culture blocks)	Yes	Liquid handling; Plate-format spectrophotometer reading of OD ₆₀₀	Generate barcode labels; Capture OD ₆₀₀ (measure of growth); System update (liquid cultures obtained)
13 Prepare glycerol stocks (long-term storage medium)	Yes	Liquid handling; freezer storage system	Generate barcode labels; Freezer organization system; System update (step completed)
14 Rearray (accepted clones)	Yes	Liquid handling; work list-based hit picking	Generate rearray files; Generate barcode labels; Generate work list; System update (storage location)

¹For bacterial gene cloning or cloning from a pooled cDNA library, a single source can be prepared and used. For cloning from unique templates, however, clones must be prepared and arrayed in a format identical to the array of PCR primers.

²In the first PCR step, gene-specific primers are used to amplify the gene and includes fixed sequences that facilitate the second PCR and/or capture steps.

³For some cloning approaches, a second PCR step with universal primers is used to add additional sequences (e.g., recombination sites) to the cloned gene insert.

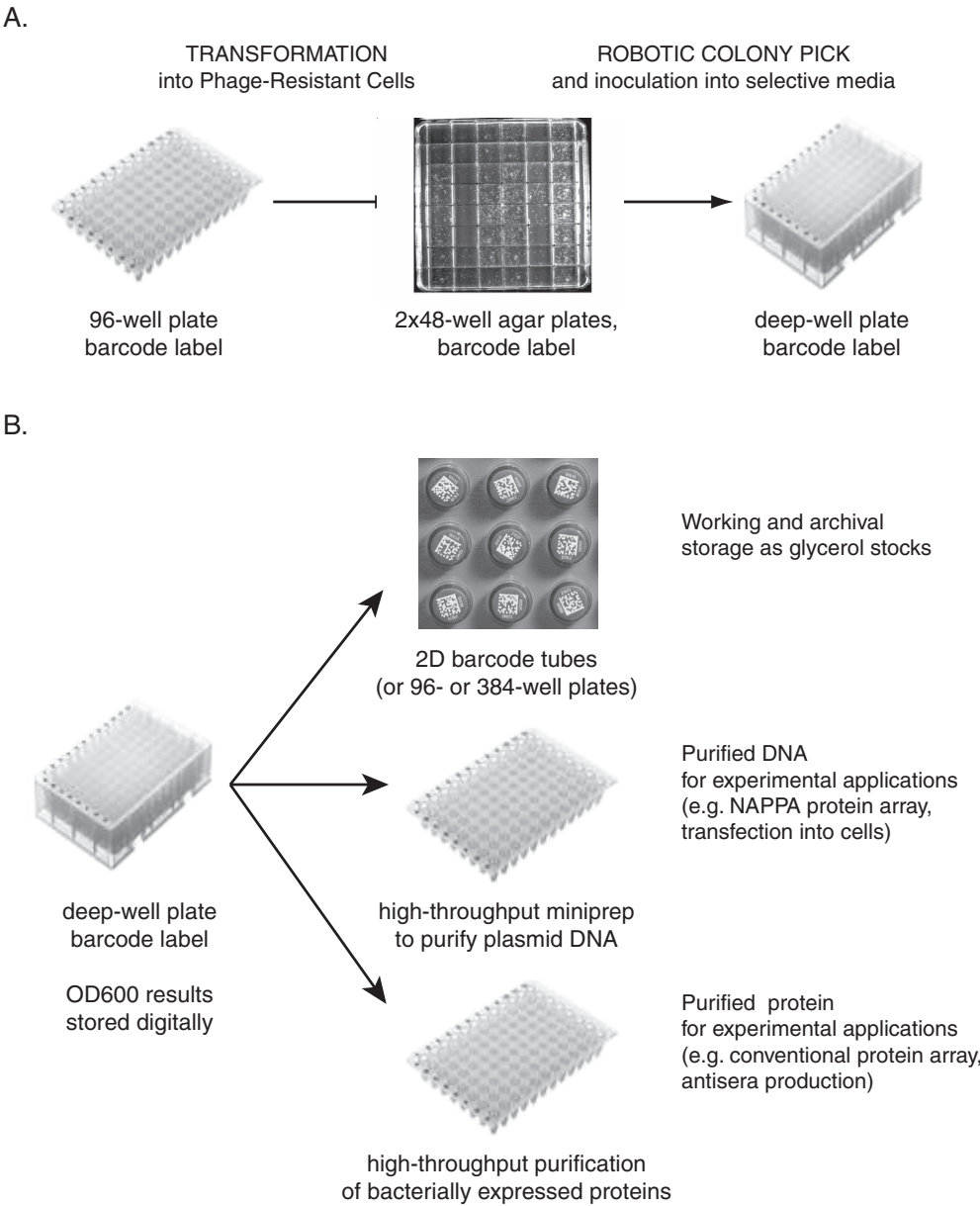


Figure 2.5 High-throughput clone production benefits from methods adaptable to robotic handling and barcode tracking. (a) Single colony isolation is automated via the use of barcode-labeled 96-well microtiter plates (left and right) and 48-well culture dishes (center). Robotic equipment scans the culture dish and then picks one colony per sector to a specific position in a deep-well culture block. (b) Liquid culture of accepted gene clones can be used to produce working or archival storage samples, and for DNA and protein production for experimental applications. Use of barcode-labeled plates and 2D barcode-labeled tubes (top center, close up of barcodes on the base of tubes) facilitates automated liquid handling, storage, and retrieval.

addition, plates used at specific steps vary in terms of material type, well depth, and material thickness in order to accommodate specific experimental requirements, including rapid temperature changes necessary for thermocycling (Table 2.3, Steps 5, 6) and larger volumes necessary for bacterial culture growth (Table 2.3, Step 12). At HIP, colony isolation is performed on specially designed 48-sector agar dishes, such that one 96-well plate can be mapped to two 48-sector dishes, and robotic equipment is designed to associate dish sectors with plate wells for automated processing (Figure 2.5).

At each step in the clone production pipeline, researchers benefit when protocols increase throughput and reduce human error via use of automation and barcode tracking (Table 2.3; Figure 2.5). A thermocycler is required for PCR amplification of DNA fragments from the template (Steps 5, 6) and a liquid handling robot (96- or 8-channel liquid handler) is required at nearly every step of the process. Specific requirements for automation include well-designed electrophoretic systems for gel isolation of amplified products (Step 7) and colony-picking robotics capable of individually addressing each sector of the 48-well agar dish (Step 11; Figure 2.6). This latter piece of equipment is particularly useful for eliminating human error and the instrument used at HIP reads barcode labels, scans for the presence of colonies, counts colonies, and selects single colonies to specific predefined wells on a deep-well culture block [Figure 2.5(a); Figure 2.6], thereby automating a process that in low-throughput pipelines is performed by hand-picking colonies with sterile toothpicks. Experience shows that even the best technicians have up to 10% error rates when processing this step by hand. Following initial development and pilot phases, laboratories with access to automated equipment can routinely produce hundreds of clones per week [15].

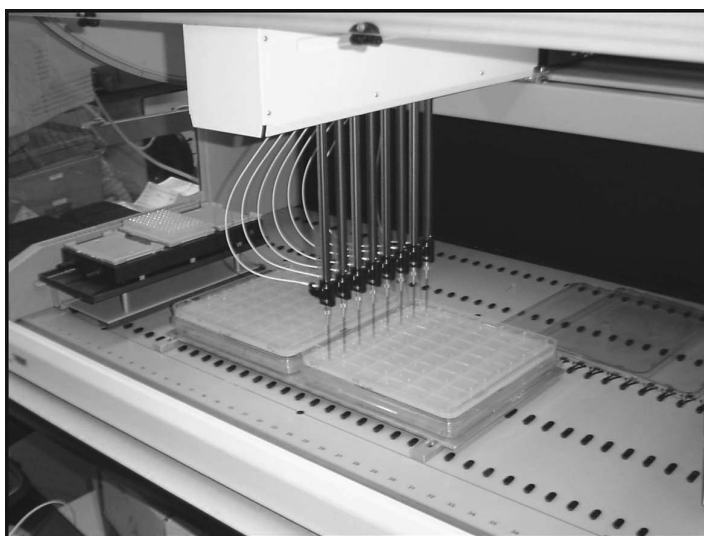


Figure 2.6 Colony selection robotics. First, plates are placed on the platform by the research technician, and barcode labels of 48-well dishes and 96-well format deep-well blocks are read by the machine and compared to a predefined work list. Next, the 48-well agar dishes are optically scanned. Finally, sterile needles are touched to single colonies from each sector of the 48-well agar dish and used to inoculate liquid medium in a deep-well culture block.

An example of successful implementation of the above strategies comes from construction of a protein-coding clone collection for the bacterium *P. aeruginosa*, which causes opportunistic infections in immuno-compromised patients and is a major threat to the health of patients with cystic fibrosis [17]. *Pseudomonas* has a predicted coding capacity of 5,570 proteins (see Section 2.2.1.1) and, based on sequence similarity, only about half of these could be assigned function. Using essentially the approach outlined in Table 2.3, researchers produced multiple clone isolates for nearly all *Pseudomonas* genes after amplification from a genomic DNA template. All steps in the production pathway relied on automation and were tracked in the FLEXGene LIMS, which also served as the initial repository database for this and other clone collections produced at HIP [51, 52].

2.2.2.3 Informatics Support for Clone Production

High-throughput clone production could not exist without reliable information tracking. Determining which information to track depends upon what types of queries will be needed later. Researchers may need to recall digital images of electrophoretic gels (to verify expected PCR fragment size), bacterial colony counts (to quantitatively assess transformation success), OD₆₀₀ (to verify bacterial growth), and plate and well histories and locations (in cases where a mismapping event is suspected). The FLEXgene LIMS, for example, tracks all stages in the clone production pipeline and provides graphical user interfaces (GUIs) to recall all relevant data (Table 2.3; [51, 52]).

This system is based upon a back-end relational database (Oracle) that tracks lineage, physical location history, and relevant associated biological information for all attempted clones. A middle layer, primarily Java and JSP, provides the business logic. The system is designed using a workflow approach. Each step in the workflow has its own logic and rules. Depending on the specific project, users can select the relevant steps to be linked together to create new workflows. The presentation layer is a web-based GUI that enables users with the correct privileges to make selections from drop-down lists, scan barcodes into the system, and upload data from instrument readers.

From start to finish, clone production steps benefit from tracking at both the plate and sample levels (Table 2.3). Unique identifiers (IDs) are assigned to physical samples and a robust barcode tracking system is used to label and track plates (Figure 2.5). The FLEXGene LIMS tracks both input and output plates, plate locations, researcher authentications, protocols used, time-stamps, and various outcomes [51, 52]. FLEXGene is also integrated with robotic instruments, such that outputs and log files are imported into FLEXGene in order to track results (Table 2.3). Clone sets cannot be advanced in the LIMS unless all steps are legitimately logged. Researchers can query the history of each clone and plate, and retrieve relevant results. They can also execute more global queries to get project summaries broken down by success per stage. Among the final steps in the clone production pathway is verifying that the clone is correct by DNA sequencing (see Section 2.2.3). All clones initiate in the “pending analysis” bin, and the goal of the validation phase is to move all clones into either the “accepted” or the “rejected” bins, the latter of which may optionally cycle back to production for repeat attempts.

2.2.3 Sequencing and Analysis

Sources for error in clone production include tracking errors at the clone and plate levels, contamination from neighboring wells, mistakes in oligonucleotide primer synthesis, and the introduction of mutations during PCR amplification. In actuality, a well-automated pipeline using well-developed chemistry results in a surprisingly low error rate at amplification. Instead, the greatest challenge in validating clones is DNA sequencing and analysis itself. DNA sequencing is the only available method to detect all of these errors but is itself a very error-prone process (much more error prone than mistakes in cloning). Because the intent for the use of these clones is to study protein function, the need for accuracy cannot be overstated (see Section 2.2). Thus, the validation of clone sets presents new challenges for sequence analysis and consequently requires the development of new methods, including software tools that automate or semiautomate the process.

2.2.3.1 Comparison of Experimental and Reference Sequences

Conceptually, the process of sequence-verifying clones is straightforward: obtain the complete sequence of each final clone and compare it to its expected reference sequence. In practice, however, analysis of sequencing results and comparison to the reference sequence present several challenges. First, individual sequencing reads often do not extend long enough to cover the entire length of the gene and thus multiple sequencing reads must be aligned and assembled to form a single consensus sequence, called a “contig.” Software that automatically aligns and assembles multiple reads is available but finicky, variably sensitive to the presence or absence of sequences at the ends of the reads, which tend to be of lower confidence. Second, the sequencing process itself is so error prone that most discrepancies between the clone sequence and its expected sequence are due to mistakes in base assignment made by the sequence analyzer, not to actual mutations in the clone sequence. To some extent this can be mitigated by careful attention to the confidence score, which is a logarithmically based score assigned to each base in a sequence read that indicates the probability that the base is correctly assigned. By this method, discrepant bases with low confidence scores are more likely to be sequencing errors than actual mistakes in the clone itself. Third, different discrepancies may result in variable consequences for the protein encoded by the gene. Some base changes are “silent” and do not affect the final protein sequence (because of the degeneracy of the genetic code), whereas others may lead to premature truncation of the protein. The decision to accept or reject a clone must be informed by the types of discrepancies and their protein consequences. Notably, this last consideration makes it important to compare sequences not only at the nucleotide level but also at the level of conceptual translation, thus adding to the complexity of the operation [15].

A common validation process strategy begins by examining the two ends of the genes using inwardly facing primers that correspond to the common plasmid vector sequences that flank each gene. The use of these “universal” primers obviates the need to obtain primers specific to the gene sequences. Comparison of “end-read” sequences to the reference confirms clone identity, thus detecting tracking and cross-contamination errors. In addition, this quick look at the clone quality can

be used to choose the best candidate(s) when production includes more than one clone per target gene (that is, >1 isolate at Step 11 in Table 2.3). Moreover, for small genes (<1 kb), end-reads may lead to a full-length contig and thus be sufficient for full analysis of the clone. Clones for which a full length contig cannot be assembled will require additional sequencing with gene-specific primers (commonly referred to as “internal” primers) in a process often referred to as a “primer walk.”

2.2.3.2 Informatics Support for Sequence Analysis

The principal tasks in sequence analysis include (1) assignment of bases (basecalling) and confidence scores to each position in trace reads generated by the sequencing instrument, (2) alignment of sequence strings to form a consensus contig along with adjustment of the confidence scores to reflect the multiple reads (usually by computing a Bayesian combination of the individual scores at the corresponding positions), (3) comparison of the assembled contigs with the reference sequence to identify discrepancies, (4) persistence of relevant data regarding each discrepancy in a database structure, and (5) decision making regarding the acceptability of clones based upon their discrepancies compared with the users’ preferences. Alignments can be done using local or global algorithms. When comparing two sequences, as in the case for discrepancy searches, global alignments such as the Needleman-Wunsch algorithm [53] are more suitable. They find the best match over the entire length of the two sequences, whereas local alignment algorithms, such as that found in BLAST, focus more on aligning segments of sequence and are more likely to result in multiple alignment units.

Sequence analysis often requires the efforts of a team of curators using visualization software that facilitates quality analysis and sequence alignment. However, this “hand-curation” process is slow and labor intensive. In an effort to automate clone sequence analyses, at least one group has developed software to automate sequence analysis. The Automated Clone Evaluation (ACE) software tool (<http://www.hip.harvard.edu/>) automates the process of sequence analysis, including matching of trace files to clone records, comparison of clone and reference sequences, detailed report of discrepancies, and automated sorting of clones into accepted or rejected categories based on user-defined criteria (Taycher et al., in preparation).

Where possible, ACE calls upon existing computational methods for base calling, confidence score assignment, sequence string alignment, contig assembly, and primer design. In addition to acting as wrapper software to these existing methods, ACE adds a number of key novel functionalities, including (1) automated methods for sorting many sequence reads from a sequencing facility into a corresponding file directory structure in which all the reads for a given clone are stored in a common directory named for that clone, which itself is located in a directory dedicated to the expected reference sequence; (2) searching for discrepancies and creating discrepancy objects that track relevant information about each discrepancy, including position, expected and actual sequence values, sequence confidence at these positions, and the protein consequences of the discrepancy; (3) aligning available clone sequence contigs with the complete reference sequence to identify sequence regions that are not represented by available sequence data (gaps) or regions of particularly poor confidence scores (low quality regions); (4) evaluating the accumulated data

for each clone to determine if the clone meets the users' criteria for acceptance or rejection, or if its outcome is pending the acquisition of further data; and (5) further dividing pending clones into categories, depending on what additional information is still needed (i.e., next steps).

The workflow for ACE begins with processing end-reads, including matching trace files to their clone records and distributing them to their specific directories. ACE then calls Phred/Phrap [54, 55] to determine the most likely base-pair at each position on the sequencing trace file ("call the bases") and set the confidence scores at each position and to assemble any clones short enough to require only end-reads. At this stage, users have the option to use ACE to rank clonal isolates if more than one isolate per gene was selected at Step 11 of the clone production process (Table 2.3), with the highest score given to clones with the best combination of high-quality match to the reference sequence and longest available sequence. ACE can then identify gaps in sequence coverage and recursively call a version of the Primer3 software tool (http://frodo.wi.mit.edu/cgi-bin/primer3/primer3_www.cgi) to generate a list of internal primers for internal sequencing of incomplete coverage (ACE "Assembly Wrapper").

Once full coverage is achieved, the ACE "Decision Tool" identifies and counts different types of discrepancies at the nucleotide and amino acid levels. Users can define threshold criteria for acceptance and rejection of clones, which can be saved as a set for future use. In some cases, the cDNA template used in the cloning project may represent a natural genetic variant of the sequence curated in RefSeq called a polymorphism (this is particularly relevant to human gene clone sets). Some users might not wish to penalize or exclude clones for harboring these polymorphisms because they represent natural variation, not cloning errors. ACE provides a tool that will compare all discrepancies with relevant GenBank sequence records to determine if there is evidence that they might be polymorphisms. In this way, discrepancies representing naturally occurring polymorphic forms can be optionally toggled to be ignored by the decision-making process for acceptance.

2.2.4 Clone Maintenance and Distribution

Once clones have been produced, verified, and accepted, both information and clone samples must be maintained over time and made available to researchers. In establishing a materials repository, there are several important considerations, but primary among them is quality control, as success rests on the basic ability to accurately deliver the clones requested. Quality control at the sample level requires careful organization and maintenance of clone samples. Clones are usually stored as glycerol stocks, a renewable resource that is fairly stable at -80°C but loses viability over time and after multiple freeze-thaw cycles. Most clone collections are stored in barcode-labeled, 96- or 384-well plates, but the state-of-the-art in the field is the use of individually barcode-labeled tubes [Figure 2.5(b)]. Use of barcode-labeled tubes has several advantages, including the ability to retrieve and thaw only the samples of interest (rather than all of the clones on a plate), thus preserving clone viability and reducing the risk of cross-contamination. Moreover, barcode-labeled tubes can be robotically arrayed in experiment-ready, user-specified formats. Emerging technologies for long-term storage of plasmid clones include

paper-based and matrix-based methods, in which bacterial culture or DNA samples are added, dried, and stored indefinitely at room temperature, followed by a hydration step for recovery.

2.2.4.1 Quality Control of Clone Information

Information about clones falls into two general categories: biological information (such as gene names, DNA sequence, and appropriate growth conditions) and storage location(s). In some cases, such as when storage location is managed by an automated storage and retrieval system that requires its own database, it may be necessary to separate the two and then integrate them using sample barcode as a unique foreign key. In this situation, the biological database enforces the relationship between sample contents and barcodes. By contrast, the storage database tracks only sample barcodes and is in a sense “indifferent” to the sample contents. For biological data storage, it is important to gain input from researchers to ensure that all relevant information is collected in order to support all anticipated queries. Moreover, it is useful to establish a dictionary of controlled vocabulary and/or draw on an external source for “official” terms (such as GO terms [33]; see Section 2.2.1.2) and to enforce the use of commonly accepted gene symbols, names, and identifiers (IDs) such as Entrez Gene IDs or organism-specific database IDs.

One example of this type of repository database is the Plasmid Information Database (PlasmID), an enterprise-class relational database that stores detailed information about clones, links clones to public databases such as NCBI, and facilitates on-line search and request of clones (<http://plasmid.hms.harvard.edu>; Zou et al., in press). In PlasmID, clones are assigned unique IDs and are associated with vector and insert information with the goal of providing relevant information and facilitating searches. With PlasmID, researchers can limit a query to genes or vectors relevant to a specific organism of interest (e.g., human or yeast) and then establish queries using gene symbols and appropriate IDs or text strings (e.g., Entrez Gene IDs).

Unique PlasmID clone IDs are linked to both two-dimensional (2D) barcode labels on the working stocks (which are in turn tracked by the automated BioBank freezer storage system) and to the separate locations of backup archival plates. The BioBank software automatically tracks all 2D barcode-labeled tubes, stores relevant data such as total storage time and frequency of access for each sample, and enables related clones (such as human kinases, which might frequently be ordered as a set) to be grouped and stored in a specific location in the freezer system. This speeds up retrieval of related clones, as robotic equipment can more quickly retrieve clones that are grouped than clones that are scattered in disparate regions of the system. Requests placed using PlasmID generate a worklist that is delivered to the BioBank robotic system for clone retrieval, integrating the two databases.

2.3 Use of Clones in Functional Proteomics Approaches

Once produced and validated, protein-coding clone collections are useful for a wide variety of experimental approaches, including large-scale study of protein function

and structure [12, 56]; identification of immunodominant and protective antigens [24, 57]; identification of protein-protein, protein-biomolecule, and protein-drug interactions [12, 58–60]; generation of functional networks and interaction maps [59, 61, 62]; and study of higher-order biological processes such as tumor progression [18]. As these increasingly available clone sets are used in high-throughput approaches, informatics and automation will have to keep pace in order for data to be collected, viewed, analyzed, and interpreted in meaningful ways.

In general, data analysis from functional proteomics requires four steps: (1) defining the experimental design, (2) capturing the data, (3) analyzing the data, and (4) integrating the data (Figure 2.7). In order to enable the relevant comparisons, database design must account for experimental design such that experimental variables (e.g., plus or minus treatment, time points, drug concentrations, etc.) are captured in a manner that can be appropriately incorporated into data analysis (Figure 2.7). Designing database architecture to be flexible enough to handle the myriad of experimental designs represents one of the most daunting challenges facing this field. Designs that appear perfectly appropriate when the first experiments are envisioned often turn out to be too limited once researchers find themselves heading in new and previously unanticipated directions.

Experimental steps may be tracked (such as via a LIMS) in order to facilitate integration of experimental protocols with automation and barcode label tracking, similar to what has been described for the clone production pipeline (see Section 2.2.2; Table 2.3). To the extent possible, it is always best if data capture can be handled automatically via direct integration of instrumentation through a network. The analysis and integration steps are often specific to the particular problem that is addressed. Below, three applications for protein-coding clone sets—protein production, protein arrays, and cell-based assays—are discussed in more detail, along with discussion of informatics support of data capture, analysis, and integration.

2.3.1 High-Throughput Protein Production

Increasingly, improved detection technologies and miniaturization have enabled biochemical protein investigation at the microgram scale. Proteins can be used in high-throughput microscale systems for identification of protein-protein interactions, enzymatic activities and substrates, and other protein-molecule interactions [9]. Purified proteins at the microscale can also be used in high-throughput screening to identify interesting targets for further investigation. Alternatively, multiple variants of the same protein can be screened for optimized characteristics like yield and solubility for selecting the best clone to scale up for structural experiments. These new applications have created a demand for methods that can produce hundreds or even thousands of proteins in small scale. Braun and colleagues [63] transferred a test set of human protein-coding clones into a bacterial expression vector that added a common in-frame epitope tag to facilitate purification. These clones were then transferred in parallel into bacteria and conditions were found that enabled robust bacterial growth (prior to turning on protein expression), efficient induction of protein production, and automated methods for protein isolation from opened bacteria. Using this approach, approximately 80% of a set of more than 300 human genes were successfully expressed and purified [63].

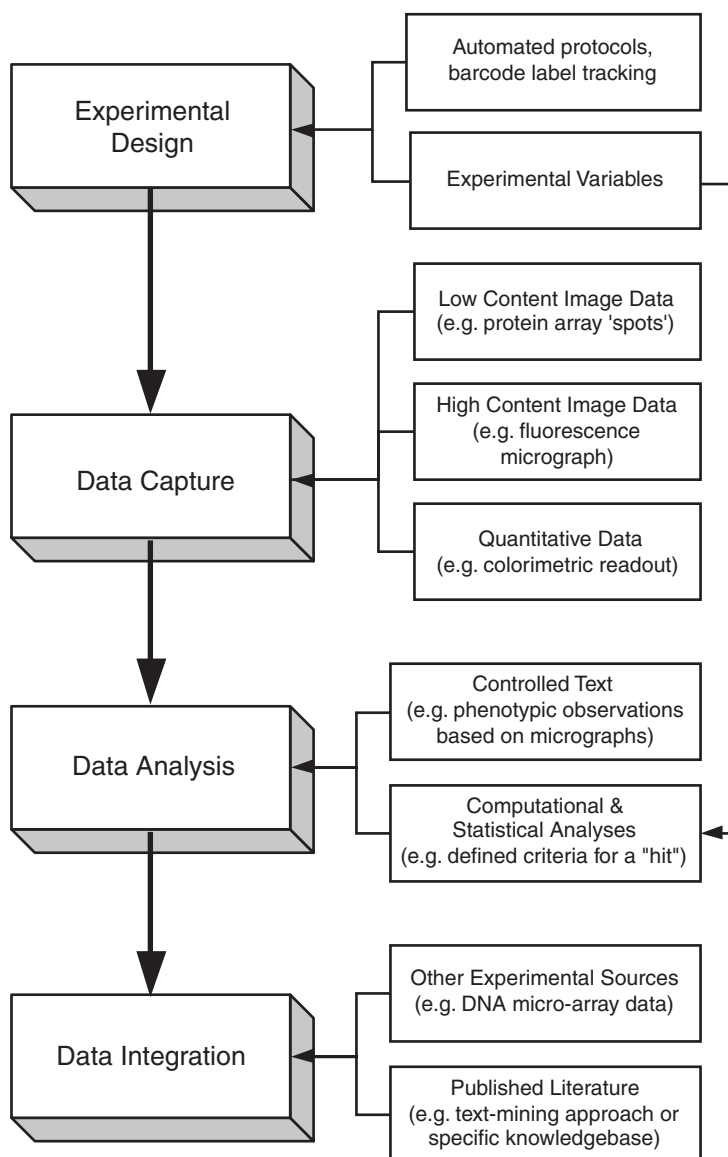


Figure 2.7 Management of information from functional proteomics-based assays can be divided into tracking of experimental design, data capture, data analysis, and data integration. Experimental design information includes tracking of individual steps and plate labels as well as capture of experimental variables used for data analysis. Data capture includes capture of raw data, such as low- or high-content images or quantitative readouts. Data analysis is used to define positive "hits" based on absolute or relative threshold levels, predefined statistical cutoffs, and/or defined qualitative criteria. Data integration can include comparison with other experimental results from within the lab (e.g., similar assays or secondary screens) or from the published literature.

2.3.1.1 Informatics Support for High-Throughput Protein Production

Critical data needed to analyze protein production include the size, purity, and molecular weight of each attempted protein. Historically, this information could be derived from separating proteins by gel electrophoresis and staining the gels with a protein avid dye like Coomassie blue. Quantifying purity necessitated scanning digital images of these gels and applying image analysis software to integrate band volumes. The addition of molecular weight and mass standards enabled size and yield prediction. The limit of this approach is about 26 proteins analyzed simultaneously. More recently automated instrumentation has become available that uses microfluidics to separate and analyze the proteins, which can handle 96 samples in parallel. These instruments inherently digitize the collected data and can provide size, yield, and purity information automatically in an output file. In both cases, data capture and analysis are often imported into a relational database, which can then be used to compare purification success with other information about the specific genes (e.g., size, hydrophobicity, known folding domains). Data integration may reveal correlations between successful protein production and these characteristics [30, 63]. Protein production is often a first step towards execution of some functional proteomics approaches (for example, for production of some types of protein arrays or for use in particular in vitro or in vivo assays). Thus data capture, analysis, and integration at the protein purification stage may be part of a broader study that requires additional informatics support, such as that outlined for protein arrays and cell-based assays below.

2.3.2 Protein Arrays

2.3.2.1 Protein Arrays Facilitate High-Throughput Analyses

Protein microarrays provide a miniaturized platform for high-throughput study of proteins and thus allow researchers to query thousands of proteins simultaneously for a given binding capacity, property, or response [64]. The different types of functional protein arrays include covalent attachment of purified proteins directly to the array surface, attachment of purified proteins via peptide tags, and, more recently, self-assembling arrays (see Section 2.3.2) (reviewed in [60]). Several properties of proteins make them challenging to array [60]. First, unlike the simple hybridization chemistry of nucleic acids, proteins demonstrate a staggering variety of chemistries, affinities, and specificities—e.g. some are hydrophobic and others hydrophilic, making it difficult to find appropriate surface chemistries for arrays. Second, proteins may require multiple copies working together, partnership with other proteins, or post-translational modification to demonstrate activity or binding. Third, there is no simple amplification process that can generate large quantities of protein. Expression and purification of proteins is often a tedious task and does not guarantee the functional integrity of the protein. Last, many proteins are notoriously unstable, raising concerns about microarray shelf life.

Ramachandran and colleagues have reported a novel approach for production of self-assembling protein microarrays: the nucleic acid programmable protein array or NAPPA approach [65]. With NAPPA, protein-coding clones in vectors appropriate for in vitro protein production are spotted to the array in a configuration

that will allow protein to be made on the array surface. Along with the protein-coding clone, an antibody that recognizes a peptide tag fused to the end of each protein (genetically added by the plasmid vector) is also printed so that as soon as protein is expressed it will be “captured” by the antibody. When the researcher is ready to use the array, an extract is added to the slides to induce protein production from each of the spotted clones. These protein arrays have been shown to be useful for identification of protein-protein interactions and will likely enable identification of protein-antibody, protein-drug, and other properties [60, 64, 65].

2.3.2.2 Informatics Support for Protein Array Analysis

For protein arrays, typical data include microscopic images of fluorescence, luminescence, or radioactivity signals across the planar surface, making data capture and analysis principally image processing steps. Unlike DNA microarrays, wherein most spots on the array will have some detectable signal, with a protein array there may only be a few spots with signal above background. After the image is captured, a digital grid is aligned to the digital image in order to associate sample information with results. This process is traditionally done by the tedious process of aligning the grid to the data manually. Automating this process is not trivial because the reality of printing instruments (e.g., slightly angled pins, unusual drying patterns) results in array features that do not always align precisely with a grid; moreover, a grid itself may be slightly askew relative to the array substrate. Recently, commercial software tools capable of detecting signal peaks and flexibly fitting them to a grid have emerged. Most of these commercial tools work to assist alignment rather than to automate it and, thus, human intervention is still demanded. Hopefully, these tools will improve to the level where this process can be fully automated.

After array features have been mapped to specific protein names, the absolute or relative values of detectable readouts can be compared to some threshold value and then can be used to define a list of positive hits. For protein array data, analysis and integration are typically done using a spreadsheet or simple in-lab database (see Section 2.3.3.3), but some groups are developing more sophisticated databases to track experimental steps, capture raw and/or analyzed data, integrate with image processing software, and automate computational analyses. At a minimum, positive hits are linked to corresponding protein-producing clones, which are then linked to gene-specific or protein-specific information in the same or in a repository database. Integration of positive hits with additional information sources can also be of help in designing follow-up experiments and interpreting results (see Section 2.3.3.3).

2.3.3 Cell-Based Functional Proteomic Assays

In addition to expression in heterologous cell systems (see Section 2.3.1), genes can also be introduced back into the organisms from which they were cloned—for example, human genes can be introduced into human cell lines—in order to test the *in vivo* effects of ectopic expression of the proteins in cells that usually produce no or low levels of the protein. Observations from this type of study are nicely complemented by classical genetic or RNA interference (RNAi)-based approaches, which effectively turn off or “dial-down” protein levels. Together, these approaches are

part of an exciting transition in the study of mammalian cell systems, where it is now possible to do comprehensive functional screens in much the same way that large-scale genetic screens are done in model organisms such as yeast, worms, and flies.

2.3.3.1 Ectopic Expression Can Be Used to Uncover Biological Function

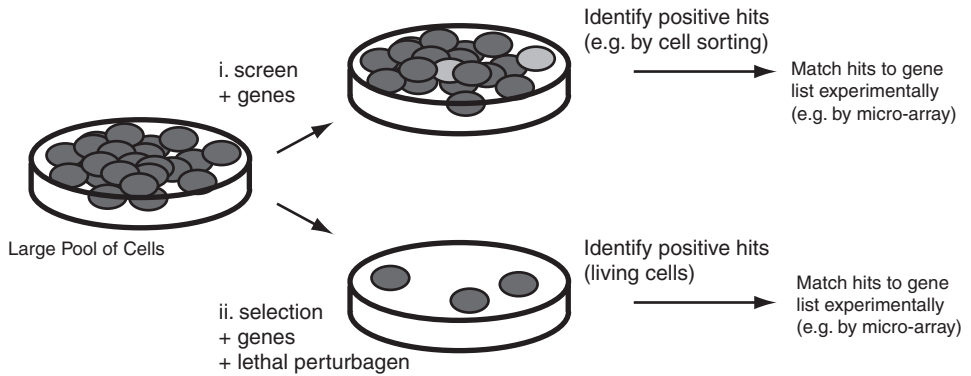
The range of phenotypes that can be detected in screens of these “perturbagens” is nearly limitless, and choices will depend on biological interest. For screens in mammalian or other tissue culture systems, the phenotypes assayed will in general fall into four basic categories: (1) establishing regulatory networks, such as addressing how each perturbagen affects the gene expression or protein-level changes of one or many other genes or proteins; (2) finding pathways important to specific behaviors, such as determining which perturbagens induce a cell-level change in viability, metabolism, mobility, morphology, or other measurable behaviors; (3) finding pathways that regulate or are regulated by external signals, for example, determining how each perturbagen affects a given cellular response, sensitivity, or resistance to an external signal like a hormone or drug; or (4) finding proteins that affect developmental processes, for example, identifying pathways that play a role in tissue-level changes such as cell outgrowth or changes in the morphology of a multicellular structure. Depending on the difficulty of the assay, it may be more or less desirable to test the maximum number of genes available, or to focus on a subset of high-likelihood informative genes.

One of the main challenges in cell-based assays is the development of quality-controlled, robust, and low-cost methods for cell screening. The number of different genes to test is large, and multiple repetitions of individual experiments are required to obtain statistically significant results. Moreover, a typical cell screening pipeline (e.g., in mammalian tissue culture) can involve a large number of liquid-handling steps, as researchers must grow bacterial cultures for each gene, purify its corresponding DNA, introduce the DNA into cells (a step that may be followed by collection of virus-containing supernatants and use of supernatants to infect a second cell type), and process the cells for phenotypic analysis. Thus, efforts are made to minimize use of expensive reagents, to reduce error, and to increase throughput. In tissue culture approaches, this will take the form of automation of liquid-handling steps at each stage of the process and, whenever possible, through use of automated detection of phenotypes. Moreover, screening approaches generally employ either a pooling and deconvolution strategy [Figure 2.8(a)], which minimizes early steps but requires additional steps after detection, or a highly parallel approach in which each perturbagen is tested individually [Figure 2.8(b)].

2.3.3.2 Informatics Support of Cell-Based Assays

Two main types of readouts result from cell-based assays such as mammalian tissue culture assays: quantitative readouts (such as levels of fluorescent, luminescent, and/or colorimetric) and digital images. The former can often be captured using automated or semiautomated plate readers that capture raw data in tab-delimited formats associated with specific 96-well plates that can be imported into a database

A. Pooling Approach to Cell-Based Assays



B. Highly Parallel Approach to Cell-Based Assays

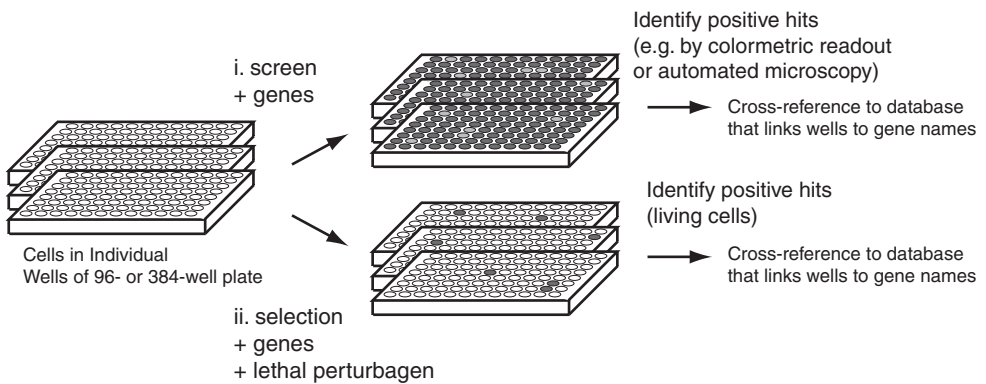


Figure 2.8 Cell-based assays can be performed using pooled or highly parallel approaches. (a) In a pooling approach, researchers introduce constructs into cells en masse and positive hits are then identified via an independent deconvolution step, such as by microarray detection of DNA barcode tags unique to each construct. Pooling has the advantage of decreasing the scale of the initial screen but the disadvantage that changes in competitive growth or other changes unrelated to the phenotype of interest may mask positive hits. (b) In a highly parallel approach, individual genes are introduced into cells in individual wells in 96- or 384-well format dishes, obviating the need for a deconvolution step. Highly parallel approaches have the advantages that each construct is tested individually and positive hits can be identified simply by referring back to an informational map that links gene and construct names to plate well positions.

and analyzed computationally. The latter, however, present additional challenges, as simple capture of a digital image alone is not sufficient to determine cell phenotypes. In many cases, trained researchers systematically analyze digital images and make observations (preferably using a predefined controlled vocabulary). State-of-the-art microscopes used for phenotypic analysis are outfitted with motorized stages and can automatically capture images from 96-well plates; in addition, some use software trained to recognize specific patterns in order to identify positive hits

[66, 67]. Clearly, the type of phenotypic readout will have an influence on informatics support and, in many cases, will require capture of digital outputs and sophisticated computational or observational results. As for protein production and protein array analyses, cell-based screens are followed by analysis and integration of data to identify a set of positive hits and help guide the course of future experimentation.

2.3.3.3 Integration of Data from Diverse Sources

The results of functional proteomics experiments are most valuable when they are put into context with other information about the same or related proteins. Indeed, integration of results with other information sources is critical in interpreting data, drawing conclusions, and planning follow-up studies. Cross-referencing of positive hits to protein or gene names is one way to link results with other information sources. However, the process is complicated by the fact that efforts to systematize gene names began fairly recently and that in the past, one, a few, or several names might have been used for the same gene. Thus even for a single organism, there can be more than one name or abbreviation for a gene and, furthermore, some gene names or abbreviations are associated with more than one gene. Compilation of data from diverse sources, then, is a significant challenge—and indeed, as the fields of genomics and proteomics progress, the task of simply “speaking the same language” so that results can be related may prove substantial. Several efforts have been made to help systematize gene names for one or more organisms. For example, a number of organism-specific resources have been trying for a long time to assign systematized gene names and/or identification numbers and, as much as possible, to enforce their use [46–48]. In addition, the NCBI Entrez Gene project has begun to systematize gene naming in a number of organisms, including for human genes, by creating unique IDs at the gene level. The use of systematized gene names, symbols, and IDs can help researchers associate positive hits from cell-based and other assays with the wealth of information that is available for some proteins.

Once an effort has been made to capture one or more systematic name, the focus can shift to the more interesting task of gathering data from other information sources. These sources include expression data; protein interaction data; evolutionary conservation and other sequence comparison information; organism-specific databases; and specifically focused “knowledge bases,” GO annotations, and other curated information sources. Even after related data have been collected, however, drawing conclusions and planning follow-up studies remain a significant challenge, at least in part because study of the function of a protein in one cell type or organism does not necessarily reveal the function of the same or related proteins in another cell type or organism. Nevertheless, experimentalists, engineers, and informaticians alike can take comfort in the idea that their cooperative efforts in the field of proteomics are likely to have a lasting impact on our understanding of the proteomes of diverse organisms, and on the fields of biology and biomedicine more generally.

References

- [1] Reeves, G. A., and J. M. Thornton, "Integrating biological data through the genome," *Hum. Mol. Genet.*, Vol. 15, Suppl. 1, 2006, pp. R81–R87.
- [2] Lander, E. S., et al., "Initial sequencing and analysis of the human genome," *Nature*, Vol. 409, 2001, pp. 860–921.
- [3] Marsden, R. L., et al., "Comprehensive genome analysis of 203 genomes provides structural genomics with new insights into protein family space," *Nucleic Acids Res.*, Vol. 34, 2006, pp. 1066–1080.
- [4] Baker, D., and A. Sali, "Protein structure prediction and structural genomics," *Science*, Vol. 294, 2001, pp. 93–96.
- [5] Gygi, S. P., et al., "Correlation between protein and mRNA abundance in yeast," *Mol. Cell Biol.*, Vol. 19, 1999, pp. 1720–1730.
- [6] Griffin, T. J., et al., "Complementary profiling of gene expression at the transcriptome and proteome levels in *Saccharomyces cerevisiae*," *Mol. Cell Proteomics*, Vol. 1, 2002, pp. 323–333.
- [7] Pandey, A., and M. Mann, "Proteomics to study genes and genomes," *Nature*, Vol. 405, 2000, pp. 837–846.
- [8] Kramer, R., and D. Cohen, "Functional genomics to new drug targets," *Nat. Rev. Drug Discov.*, Vol. 3, 2004, pp. 965–972.
- [9] Braun, P., and J. LaBaer, "High throughput protein production for functional proteomics," *Trends Biotechnol.*, Vol. 21, 2003, pp. 383–388.
- [10] Abbott, A., "How to spot a protein in a crowd," *Nature*, Vol. 402, 1999, pp. 716–717.
- [11] Gygi, S. P., et al., "Evaluation of two-dimensional gel electrophoresis-based proteome analysis technology," *Proc. Natl. Acad. Sci. USA*, Vol. 97, 2000, pp. 9390–9395.
- [12] Phizicky, E., et al., "Protein analysis on a proteomic scale," *Nature*, Vol. 422, 2003, pp. 208–215.
- [13] Joyce, A. R., and B. O. Palsson, "The model organism as a system: integrating 'omics' data sets," *Nat. Rev. Mol. Cell Biol.*, Vol. 7, 2006, pp. 198–210.
- [14] Temple, G., et al., "From genome to proteome: developing expression clone resources for the human genome," *Hum. Mol. Genet.*, Vol. 15, Suppl. 1, 2006, pp. R31–R43.
- [15] Pearlberg, J., and J. LaBaer, "Protein expression clone repositories for functional proteomics," *Curr. Opin. Chem. Biol.*, Vol. 8, 2004, pp. 98–102.
- [16] Rual, J. F., D. E. Hill, and M. Vidal, "ORFeome projects: gateway between genomics and omics," *Curr. Opin. Chem. Biol.*, Vol. 8, 2004, pp. 20–55.
- [17] LaBaer, J., et al., "The *Pseudomonas aeruginosa* PA01 gene collection," *Genome Res.*, Vol. 14, 2004, pp. 2190–2200.
- [18] Witt, A. E., et al., "Functional proteomics approach to investigate the biological activities of cDNAs implicated in breast cancer," *J. Proteome Res.*, Vol. 5, 2006, pp. 599–610.
- [19] Aguiar, J. C., et al., "High-throughput generation of *P. falciparum* functional molecules by recombinational cloning," *Genome Res.*, Vol. 14, 2004, pp. 2076–2082.
- [20] Lamesch, P., et al., "C. *elegans* ORFeome version 3.1: increasing the coverage of ORFeome resources with improved gene predictions," *Genome Res.*, Vol. 14, 2004, pp. 2064–2069.
- [21] Reboul, J., et al., "C. *elegans* ORFeome version 1.1: experimental verification of the genome annotation and resource for proteome-scale protein expression," *Nat. Genet.*, Vol. 34, 2003, pp. 35–41.
- [22] Dricot, A., et al., "Generation of the *Brucella melitensis* ORFeome version 1.1," *Genome Res.*, Vol. 14, 2004, pp. 2201–2206.
- [23] Hudson, J. R., Jr., et al., "The complete set of predicted genes from *Saccharomyces cerevisiae* in a readily usable form," *Genome Res.*, Vol. 7, 1997, pp. 1169–1173.

- [24] McKevitt, M., et al., "Systematic cloning of *Treponema pallidum* open reading frames for protein expression and antigen discovery," *Genome Res.*, Vol. 13, 2003, pp. 1665–1674.
- [25] Brent, M. R., "Genome annotation past, present, and future: how to define an ORF at each locus," *Genome Res.*, Vol. 15, 2005, pp. 1777–1786.
- [26] Winsor, G. L. et al., "*Pseudomonas aeruginosa* genome database and pseudoCAP: facilitating community-based, continually updated, genome annotation," *Nucleic Acids Res.*, Vol. 33, 2005, pp. D338–D343.
- [27] *C. elegans*, Sequencing Consortium, "Genome sequence of the nematode *C. elegans*: a platform for investigating biology," *Science*, Vol. 282, 1998, pp. 2012–2018.
- [28] M. D. Adams, et al., "The genome sequence of *Drosophila melanogaster*," *Science*, Vol. 287, 2000, pp. 2185–2195.
- [29] International Human Genome Sequencing Consortium, "Finishing the euchromatic sequence of the human genome," *Nature*, Vol. 431, 2004, pp. 931–945.
- [30] R. D. Finn, et al., "Pfam: clans, web tools and services," *Nucleic Acids Res.*, Vol. 34, 2006, pp. D247–D251.
- [31] A. Krogh, et al., "Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes," *J. Mol. Biol.*, Vol. 305, 2001, pp. 567–580.
- [32] Hirokawa, T., S. Boon-Chieng, and S. Mitaku, "SOSUI: classification and secondary structure prediction system for membrane proteins," *Bioinformatics*, Vol. 14, 1998, pp. 378–379.
- [33] Gene Ontology Consortium, et al., "The Gene Ontology (GO) project in 2006," *Nucleic Acids Res.*, Vol. 34, 2006, pp. D322–D326.
- [34] Park, J., et al., "Building a human kinase gene repository: bioinformatics, molecular cloning, and functional validation," *Proc. Natl. Acad. Sci. USA*, Vol. 102, 2005, pp. 8114–8119.
- [35] Huang, M., et al., "Discovering patterns to extract protein-protein interactions from full texts," *Bioinformatics*, Vol. 20, 2004, pp. 3604–3612.
- [36] Daraselia, N., et al., "Extracting human protein interactions from MEDLINE using a full-sentence parser," *Bioinformatics*, Vol. 20, 2004, pp. 604–611.
- [37] Korbel, J. O., et al., "Systematic association of genes to phenotypes by genome and literature mining," *PLoS Biol.*, Vol. 3, 2005, p. e134.
- [38] Tiffin, N., et al., "Integration of text- and data-mining using ontologies successfully selects disease gene candidates," *Nucleic Acids Res.*, Vol. 33, 2005, pp. 1544–1552.
- [39] Hu, Y., et al., "Analysis of genomic and proteomic data using advanced literature mining," *J. Proteome Res.*, Vol. 2, 2003, pp. 405–412.
- [40] Shah, P. K., et al., "Extraction of transcript diversity from scientific literature," *PLoS Comput. Biol.*, Vol. 1, 2005, pp. e10.
- [41] Yuan, X., et al., "An online literature mining tool for protein phosphorylation," *Bioinformatics*, Vol. 22, 2006, pp. 1668–1669.
- [42] Hu, Z. Z., et al., "Literature mining and database annotation of protein phosphorylation using a rule-based system," *Bioinformatics*, Vol. 21, 2005, pp. 2759–2765.
- [43] Yuryev, A., et al., "Automatic pathway building in biological association networks," *BMC Bioinformatics*, Vol. 7, 2006, p. 171.
- [44] Hu, Y., and J. Labaer, "Tracking gene-disease relationships for high-throughput functional studies," *Surgery*, Vol. 136, 2004, pp. 504–510.
- [45] LaBaer, J., "Mining the literature and large datasets," *Nat. Biotechnol.*, Vol. 21, 2003, pp. 976–977.
- [46] Grumblin, G., and V. Strelets, "FlyBase: anatomical data, images and queries," *Nucleic Acids Res.*, Vol. 34, 2006, pp. D484–D488.
- [47] Schwarz, E. M., et al., "WormBase: better software, richer content," *Nucleic Acids Res.*, Vol. 34, 2006, pp. D475–D478.

- [48] Christie, K. R., et al., “*Saccharomyces* Genome Database (SGD) provides tools to identify and analyze sequences from *Saccharomyces cerevisiae* and related sequences from other organisms,” *Nucleic Acids Res.*, Vol. 32, 2004, pp. D311–D314.
- [49] Dwight, S. S., et al., “*Saccharomyces* genome database: underlying principles and organization,” *Brief Bioinform.*, Vol. 5, 2004, pp. 9–22.
- [50] Marsischky, G., and J. LaBaer, “Many paths to many clones: a comparative look at high-throughput cloning methods,” *Genome Res.*, Vol. 14, 2004, pp. 2020–2028.
- [51] Brizuela, L., P. Braun, and J. LaBaer, “FLEXGene repository: from sequenced genomes to gene repositories for high-throughput functional biology and proteomics,” *Mol. Biochem. Parasitol.*, Vol. 118, 2001, pp. 155–165.
- [52] Brizuela, L., et al., “The FLEXGene repository: exploiting the fruits of the genome projects by creating a needed resource to face the challenges of the post-genomic era,” *Arch. Med. Res.*, Vol. 33, 2002, pp. 318–324.
- [53] Needleman, S. B., and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *J. Mol. Biol.*, Vol. 48, 1970, pp. 443–453.
- [54] Ewing, B., and P. Green, “Base-calling of automated sequencer traces using phred. II. Error probabilities,” *Genome Res.*, Vol. 8, 1998, pp. 186–194.
- [55] Ewing, B., et al., “Base-calling of automated sequencer traces using phred. I. Accuracy assessment,” *Genome Res.*, Vol. 8, 1998, pp. 175–185.
- [56] Zhu, H., et al., “Global analysis of protein activities using proteome chips,” *Science*, Vol. 293, 2001, pp. 2101–2105.
- [57] Maignani, V., R. Rappuoli, and M. Pizza, “Reverse vaccinology: a genome-based approach for vaccine development,” *Expert Opin. Biol. Ther.*, Vol. 2, 2002, pp. 895–905.
- [58] Suter, B., D. Auerbach, and I. Stagljar, “Yeast-based functional genomics and proteomics technologies: the first 15 years and beyond,” *Biotechniques*, Vol. 40, 2006, pp. 625–644.
- [59] Uetz, P., and R. E. Hughes, “Systematic and large-scale two-hybrid screens,” *Curr. Opin. Microbiol.*, Vol. 3, 2000, pp. 303–308.
- [60] LaBaer, J., and N. Ramachandran, “Protein microarrays as tools for functional proteomics,” *Curr. Opin. Chem. Biol.*, Vol. 9, 2005, pp. 14–19.
- [61] Gavin, A. C., et al., “Functional organization of the yeast proteome by systematic analysis of protein complexes,” *Nature*, Vol. 415, 2002, pp. 141–147.
- [62] Li, S., et al., “A map of the interactome network of the metazoan *C. elegans*,” *Science*, Vol. 303, 2004, pp. 540–543.
- [63] Braun, P., et al., “Proteome-scale purification of human proteins from bacteria,” *Proc. Natl. Acad. Sci. USA*, Vol. 99, 2002, pp. 2654–2659.
- [64] MacBeath, G., and S. L. Schreiber, “Printing proteins as microarrays for high-throughput function determination,” *Science*, Vol. 289, 2000, pp. 1760–1763.
- [65] Ramachandran, N., et al., “Self-assembling protein microarrays,” *Science*, Vol. 305, 2004, pp. 86–90.
- [66] Zhou, X., et al., “Towards automated cellular image segmentation for RNAi genome-wide screening,” *Int. Conf. Med. Image Comput. Comput. Assist. Interv.*, Vol. 8, 2005, pp. 885–892.
- [67] Echeverri, C. J., and N. Perrimon, “High-throughput RNAi screening in cultured cells: a user’s guide,” *Nat. Rev. Genet.*, Vol. 7, 2006, pp. 373–384.

PART II

Analysis: Signal Processing

Introduction to Biological Signal Processing at the Cell Level

Maya R. Said

There is tremendous promise at the intersection of signal processing and biology. Signal processing can contribute sophisticated models for understanding biological systems, while biology has a great repertoire of novel algorithms and filtering techniques to contribute to signal processing. We define “biological signal processing” as the discipline aimed at both (a) understanding and modeling the biological algorithms implemented by living systems using signal processing theory, and (b) the efforts seeking to use biology as a metaphor to formulate novel signal processing algorithms for engineered systems [1]. The first objective (a) refers to focusing on biological systems as an endpoint and using signal processing as a tool for better understanding, while in the second objective (b) the focus is on signal processing as an end objective in and of itself, with biological systems representing tools and metaphors. It is important to note the duality between these two objectives. In other words, it is through using signal processing to model biological signaling that one would be able to develop novel signal processing algorithms based on biological signaling. This chapter presents an introduction to biological signal processing *at the cell level*, providing an overview of the underlying concepts and highlighting some case examples. Given the introductory nature of this chapter, the focus here is on objective (a) of the definition, that is, on using signal processing to model and understand biological algorithms within cells. We therefore present examples where signal processing techniques have been used to develop insight into biological systems.

The interplay between signals, systems, and biology has a long and rich history. Documented accounts date back to the nineteenth century with the work of Claude Bernard citing a collection of interrelated biological regulators and introducing the concept of homeostasis: the process by which living organisms maintain a constant internal state in the face of environmental challenges [2]. Ludwig von Bertalanffy, one of the most important theoretical biologists of the first half of the twentieth century, defined General System Theory in his 1968 book on the topic by noting that “there appear to exist general system laws which apply to any system of a particular type, irrespective of the particular properties of the systems and the elements

involved (. . .) These considerations lead to the postulate of a new scientific discipline which we call general system theory. Its subject matter is formulation of principles that are valid for ‘systems’ in general, whatever the nature of the component elements and the relations or ‘forces’ between them” [3]. A few years earlier, the renowned mathematician, Norbert Wiener, published his book on cybernetics [4], describing the process of control guided by feedback through communication of information, which is critical to both living and nonliving systems. A number of systems books published in the early 1960s further described the interplay between biology and signals and systems. Grodins’ 1963 book was aimed at demonstrating the power and usefulness of the systems approach in advancing our understanding of complex biological regulators [5]. Specifically, he considered two examples of biological systems: the respiratory system and the cardiovascular system. A systems view of molecular biology was considered by Reiner in the late 1960s [6]. Interestingly, in his book he argues that having a clear idea of how a living organism works is not an automatic consequence of knowing everything about its DNA, repressors and inducers, and enzymes—an observation that resonates particularly well in the postgenomic era of the twenty-first century, as scientists and engineers are finding that knowing the blueprint of a living system (its DNA sequence) is not sufficient to understanding its function.

While the interplay between signal processing and biology is not a new one, it is becoming critical, now more than ever, to strengthen it as technological advances and breakthroughs in molecular biology are giving us access to a wealth of system-level biological information in a field that has historically focused on a “reductionist” component-level view. In fact, although the molecular components comprising cells are being cataloged at a continually accelerating rate, there is no effective knowledge of how these components work together as an integrated dynamic system to yield output cell functions, such as survival or proliferation, as responses to information, such as chemical or mechanical signals, presented in the cell environment. Understanding how cells do signal processing therefore requires models that define layers of abstractions in order to view signaling algorithms at different resolutions. Such a systems approach is at the heart of the field of signal processing and therefore it is expected that meaningful contributions can be made by applying a signals and systems approach to problems in molecular biology. In the next sections we provide some case examples of such contributions with the hope that these may spark the reader’s interest to explore further directions at the intersection of signal processing and biology. The choice of examples is motivated by the different signal processing concepts they highlight. For example, examining the problem of DNA sequencing provides an example of how signal detection and estimation techniques such as Wiener filtering and homomorphic blind deconvolution are applied, while the protein signaling example illustrates the use of singular value decomposition and system function modeling as techniques for system identification and analysis.

It is important to keep sight of our overall objective, which is to understand how biological systems work holistically. With this in mind, this chapter is structured as follows: after introducing some signal processing concepts in Section 3.1, we gradually progress through the layers of biological information starting with signal detection and estimation in Section 3.2 using DNA, genes, and proteins signals, progressing to systems identification and analysis within cells in Section 3.3, where we investigate gene regulation and protein signaling systems. We conclude in Sec-

tion 3.4 with a summary and a brief discussion of novel signal processing techniques inspired by biological systems.

3.1 Introduction to Fundamental Signal Processing Concepts

Signal processing is a well-established and constantly evolving field. The generality of signal processing concepts and their applicability to problems spanning different engineering and scientific areas has generated great interest and has led to many years of productive research and important results. In particular, a vast number of books, chapters, and papers have been written on the topic and present excellent background as well as in-depth application-specific analyses. In this section, we review fundamental signal processing concepts relevant to biological applications. In particular, we highlight definitions and techniques that will be used in the case examples we consider later in the chapter. For a comprehensive introduction to the field of signal processing, the reader is referred to [7] and [8]. In the following, we start by reviewing signals abstractions and representations. We then introduce the concept of systems and present methods to analyze them. We conclude with a discussion of random processes and spectral analysis.

3.1.1 Signals

In signal processing theory, signals are abstracted by mathematical functions of independent variables. These functions could arise in both natural environments as well as synthesized designs. For example, natural speech could be abstracted by a function where the independent variable is time and the values taken by the function correspond to acoustic pressure. A speech signal could also be synthesized by generating a mathematical function of time and changing the acoustic pressure according to the function values. Note that in this case, the signal can refer to either the speech itself or the mathematical function representing it. This is generally the case in signal processing textbooks, where the term signal is used interchangeably to refer to the physical phenomenon or the functional abstraction. A computer image could be abstracted as a signal where the independent variable is spatial location and the function corresponds to the pixel value at that location. Other examples include electrical signals such as voltage as a function of time, mechanical signals such as force as a function of space or time, and chemical signals such as concentration as a function of space or time. Biological signals span different forms, ranging from electrical signals such as voltage variation through nerve synapses, physical signals such as mechanical stress or pressure at the surface of cells, to chemical signals such as hormone concentrations in the bloodstream. The first step in applying signal processing techniques to biology is to clearly define the signals we wish to study by identifying the independent variables and functions of interest.

3.1.1.1 “Time”-Domain Representation

Much of modern signal processing evolved from the field of time series analysis [9]. As a result signals of primary interest to signal processing were time-series, i.e., functions of time. The term “time-domain representation” therefore emerged to

refer to this natural view of signals in contrast to mathematical transformations of signals such as the frequency-domain representation discussed later. It is important to note, however, that although the term “time-domain” has become standard in signal processing, “time” is defined loosely and often refers to any one-dimensional independent variable of interest. This is particularly important when dealing with biological signals, since on many occasions the independent variable is not related to temporal information. The time-domain representation of a signal therefore refers to the representation that naturally led to the signal abstraction, specifically in relation to the independent variable. For example, for the measured speech signal, the time-domain representation is the function of time. For a one-dimensional spring element, the “time” variable is displacement and the time-domain representation is the function of displacement. Typically independent variables for most signals traditionally encountered in signal processing problems are either time or displacement/space; however, sometimes the independent variable may correspond to a different dimension or measurement. For example, as we later discuss, for gene signals the independent variable is the base-pair position and the “time-domain” representation corresponds to the function of the base-pair position. For protein signals, the independent variable is amino-acids position. It is therefore critical to maintain the generality of the meaning of time, especially when considering biological problems.

Independent variables, and therefore the corresponding signals, can be divided into two broad classes: continuous and discrete. Continuous variables can take on a continuum of values, while discrete variables can take on only a discrete set of values. Signals that represent functions of continuous variables are referred to as continuous-time signals, while signals representing functions of discrete variables are referred to as discrete-time signals. Examples of continuous-time signals include speech and chemical concentrations, while digital images, DNA, and protein sequences represent discrete-time signals. In this chapter, we refer to the continuous variable as t and to the corresponding continuous-time signal as $x(t)$, while the discrete variable is referred to as n and the corresponding discrete-time signal as $x[n]$. In addition, the value of the function can either be discrete or continuous, corresponding to discrete or analog signals respectively. Digital signals are a subset of discrete signals where the function takes on binary values.

3.1.1.2 Frequency-Domain Representation

Signal transformations correspond to alternative mathematical representations leading to additional insight and understanding of the signal of interest. Transforms are useful for a number of applications, including enhancement, feature detection, and compression. Examples of transformations that have been widely used in signal processing include the Fourier transform and its generalization into the Laplace and Z transforms, the discrete Fourier transform, the cosine transform, the short-time Fourier transform, and the wavelet transform. Here, we briefly introduce the Fourier transform, the discrete Fourier transform, and the short-time Fourier transform. The reader is referred to [7, 8, 10] for more detailed discussions of the different transforms.

The Fourier transform is a frequency-domain representation of a signal. In general, the frequency domain corresponds to a representation of a signal as a linear

combination of periodic signals with varying frequencies. The periodic signals in the Fourier transform are complex exponentials. Specifically, the Fourier transform of a continuous-time signal $x(t)$ is denoted by $X(\omega)$ and defined as

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t} dt \quad 3.1$$

The time signal can be recovered from the Fourier transform by a similar operation:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega)e^{j\omega t} d\omega \quad 3.2$$

The Fourier transform pair of a discrete-time signal $x[n]$ is given by

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \quad 3.3$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(e^{j\omega})e^{j\omega n} d\omega \quad 3.4$$

Note that the Fourier transform of a discrete signal is continuous and periodic with period 2π . It is often plotted for only one period, since it captures the entire frequency content of the signal.

When a signal lends itself to Fourier analysis, the frequency domain representation often provides additional insights about the underlying signal that are otherwise hard to see. Fourier transforms have a number of properties resulting from the complex exponential basis functions. These properties as well as Fourier transforms of different functions are discussed and tabulated in most signal processing textbooks and therefore are not included here. While for most signals obtained from experimental measurements, the Fourier transform is obtained through mathematical manipulations of the original signal, there are cases where the direct measurement actually occurs in the frequency domain. An example of such a case is X-ray crystallography, where the diffraction pattern measured is the magnitude of the Fourier transform of the electron density of the crystal and therefore recovering the signal involves taking an inverse Fourier transform.¹

The discrete Fourier transform (DFT) can be thought of as a uniform sampling of the Fourier transform. It is defined for discrete finite inputs and is widely used, since most measured signals are finite in nature, and it provides a means to efficiently implement digital signal processing algorithms. The DFT, $X[k]$, of a finite signal $x[n]$, where $0 \leq n \leq N-1$ is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \quad 3.5$$

The corresponding synthesis equation is given by:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j2\pi kn/N} \quad 3.6$$

1. The absence of phase information makes this problem nontrivial.

It should be clear from comparing (3.5) with (3.3) that the DFT of the finite signal $x[n]$ is equal to samples of the Fourier transform of $x[n]$ at $\omega = 2\pi k/N$. As we discuss later in this chapter, since many biological signals we consider are discrete and finite, the DFT tends to be the transform of choice for frequency analysis in these applications.

Fourier transforms provide frequency information for the entire time signal, which is an accurate representation when signals are stationary. In other words, signals are decomposed into a sum of complex exponentials with fixed coefficients, that is, it is assumed that the frequencies and amplitudes of the individual sines and cosines do not change with time. However, for many signals of interest a concept of time-varying frequency content is more appropriate. For these signals, the amplitudes and frequencies may change over time, and therefore we are interested in studying their frequency content as it evolves in time. Examples of such signals include speech and radar signals, as well as many biological signals, as we later illustrate in the chapter. The disadvantage of applying the Fourier transform to these signals is that it does not provide frequency-time resolution, that is, it provides information about frequencies in a signal without providing information about their time localization. The short-time Fourier transform addresses this issue by continuously recomputing the Fourier transform of a sequence as it is passed through a sliding window. More precisely, the short-time Fourier transform of $x[n]$ is defined by:

$$X[n, \lambda] = \sum_{m=-\infty}^{\infty} x[n+m]w[m]e^{-i\lambda m} \quad 3.7$$

where $w[m]$ is a window sequence (e.g., a length L rectangular window where $w[m] = 0$ outside the interval $0 \leq m \leq L - 1$ for some integer L) and λ is the frequency. A detailed discussion of the short-time Fourier transform can be found in a number of textbooks such as [11, 12].

The wavelet transform provides another time-frequency representation that has proven to be useful in many applications, including biological applications. The reader is referred to [10] for an overview of this transform.

It should be noted that there is an inherent tradeoff between time localization and frequency localization, that is, resolving the exact frequency as well as the exact time of occurrence of this frequency in a signal is not possible.

3.1.1.3 Multidimensional Signals

Most of our discussion so far focused on signals that are functions of one independent variable, i.e., one-dimensional signals. Often, however, we may be interested in signals that span many dimensions. Most of the concepts discussed in the previous sections easily expand for multidimensional signals. An excellent overview of multidimensional signal processing techniques is provided in [13].

3.1.2 Systems

In the previous section we referred to signals as mathematical abstractions of functions of independent variables. Similarly, systems are abstracted by mathematical transformations of these functions. Specifically, a system transforms an input signal

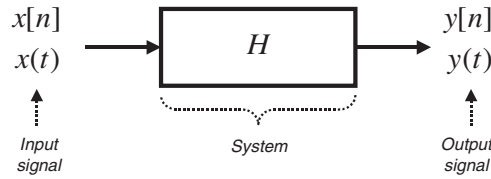


Figure 3.1 A system is defined as a mapping of an input signal into an output signal.

into an output signal as shown in Figure 3.1. There is a wide variety of systems, including ones found in nature, those engineered by humans, as well as mathematical constructs. For example, the transform domain representation discussed in the previous section can be considered as a system that takes in a signal and transforms it into its Fourier domain representation. An electrical circuit or a CD player is an example of a system that takes in a voltage or a digital signal and transforms it into another voltage or an audio signal respectively. Biological systems are examples of natural systems. For example, the ear can be modeled as a system that takes as input audio waves and transforms them into brain signals. As we see later in this chapter, genetic and protein networks are examples of systems that take in a biological input signal (or a multitude of input signals) and transform it into another biological signal (or a multitude of biological output signals).

Systems can be categorized as continuous-time systems, which take as input a continuous-time signal and output a continuous-time signal, or as discrete-time systems, whose inputs and outputs are discrete-time signals. Another class of systems that may be of interest includes hybrid systems, which deal with a mixture of continuous-time and discrete-time signals.

A special class of systems is the class of linear and time-invariant (LTI) systems. LTI systems have special properties that make their analysis tractable and, therefore, significant insight about these systems can be developed. While most real world systems, including biological systems, are not LTI, many systems can be approximated as linear and time invariant for a small enough input range. LTI techniques can therefore be used to develop insight about these systems.

3.1.2.1 LTI Systems

A system is considered linear if a weighted sum of input signals leads to the corresponding weighted sum, using the same original weights, of the corresponding output signals. Specifically let $x_1[n]$ and $x_2[n]$ be two input signals and $y_1[n]$ and $y_2[n]$ the corresponding outputs. Now consider the input $x_3[n] = ax_1[n] + bx_2[n]$ to the system, where a and b are real. The system is considered linear if and only if the corresponding output $y_3[n]$ is given by $y_3[n] = ay_1[n] + by_2[n]$. A system is time invariant if the output of a time-shifted version of the input signal corresponds to the time-shifted version of the output signal. Specifically, let $x_4[n] = x_3[n - n_0]$, where n_0 is any integer. Then, a system is time invariant if and only if the corresponding output $y_4[n]$ is given by $y_4[n] = y_3[n - n_0]$. Similar expressions apply for continuous time. A linear time-invariant (LTI) system is defined as a system that is both linear and time invariant.

The superposition and shift invariance properties of LTI systems allow them to be completely characterized by their impulse response, that is, the response of the system to a unit impulse. As a result, once the impulse response is identified, one can compute the output to any input signal. Specifically, let $h(t)$ and $h[n]$ denote the impulse responses of a continuous-time and discrete-time LTI system respectively, and let $x(t)$ and $x[n]$ denote the corresponding inputs; the outputs are given by the convolution of the inputs with the corresponding impulse response as follows:

$$y(t) = x(t) * h(t) = \int_{\tau=-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \quad 3.8$$

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \quad 3.9$$

An important property of LTI systems is that complex exponentials are eigenfunctions of these systems, that is, if the input is a complex exponential, the output is a scaled version of that complex exponential. A direct consequence of this property is that convolution in the time domain corresponds to multiplication in the frequency domain, which greatly simplifies the analysis of LTI systems. For the systems described in (3.8) and (3.9) above, the corresponding Fourier domain expressions are

$$Y(\omega) = X(\omega) \times H(\omega) \quad 3.10$$

$$Y(e^{j\omega}) = X(e^{j\omega}) \times H(e^{j\omega}) \quad 3.11$$

where $Y(\omega)$, $X(\omega)$, $H(\omega)$ are the Fourier transforms of the continuous-time variables $y(t)$, $x(t)$, $h(t)$, and $Y(e^{j\omega})$, $X(e^{j\omega})$, $H(e^{j\omega})$ are the Fourier transforms of the discrete-time variables $y[n]$, $x[n]$, $h[n]$, respectively.

Convolution is commutative, associative, and distributive. As a result, a series interconnection of systems can be represented as one system with impulse response equal to the convolution of the individual impulse responses of the interconnected systems. A parallel interconnection of systems, on the other hand, has an impulse response equal to the sum of the individual impulse responses. These properties are very useful when considering interconnections of multiple systems as well as system analysis at varying degrees of resolution. This is particularly relevant in many biological problems, since the goal in many of these problems is not only to reverse engineer the underlying system but to also understand its modular architecture and identify design components at different molecular resolutions.

3.1.2.2 Nonlinear System Theory

Signal processing has traditionally focused on linear systems due to the richness and tractability of their mathematical tools, the many practical applications where they can be used, as well as the insight they provide. However, linear systems do not always provide adequate representations of natural systems and therefore nonlinear approaches are sometimes required to model specific phenomena and to analyze

and represent systems of interest. Examples of nonlinear systems include chaotic dynamic systems and quantizers.

Since nonlinearity is defined by the absence of a property (linearity), the field of nonlinear signal processing is very broad and includes different classes of nonlinear systems as well as methodologies. We do not attempt to survey the field here. Instead, we introduce later in this chapter a number of nonlinear techniques in the context of specific examples that have proved to be useful in dealing with biological problems. Examples of such techniques include homomorphic signal processing applied to DNA basecalling and Markov-modulated Markov chains applied to protein signaling. The interested reader is referred to the extensive literature on nonlinear signal processing for more details on the different methodologies and approaches. An introduction and overview of the topic is presented in a number of textbooks such as [14–16].

3.1.3 Random Processes and Spectral Analysis

So far, the discussion has focused on deterministic signals, that is, on signals that, in principle, are perfectly predictable and whose values are readily available or measurable. However, in some applications, we may not have precise knowledge of the value of a signal at a given time or how it evolves. Such situations may arise due to imprecisions in the measurement techniques or the inability to perfectly measure the signal, inherent fluctuations in the source of the signal, as well as imperfect knowledge of the medium over which a deterministic signal is transmitted. In these situations, the signals of interest are random, that is, their values cannot be measured or predicted perfectly (or alternatively signals that, upon repeated measurements, lead to different values).

A random process is a collection of random variables, one for each time point. A random signal is a realization of a random process. In other words, it is a collection of samples that correspond to particular outcomes of the random variables. Random processes are fully characterized by the individual and joint probability distributions of all the underlying random variables. Random processes are very useful mathematical abstractions. They allow us to reason with uncertainty. In addition, they provide a framework that allows us to focus on the common features of a collection of signals. In particular, sometimes we choose to model a collection of deterministic signals as a random process to identify and focus on the similarities among them rather than the distinct features that make them different. Examples include speech processing techniques, which have been successful primarily due to the fact that they model the speech signal as a random process. Speech is clearly not random in the sense of unpredictability. However, modeling a speech signal as a realization of a random process allows us to extract features inherent to all speech signals and therefore provides important information for speech processing and synthesis. In this section, we provide brief highlights of some of the results that are useful when dealing with random processes. A detailed treatment of the theory of random processes and statistical signal processing can be found in a variety of excellent texts such as [17–20].

Obtaining the full statistical characterization of a random process can be very difficult and sometimes impossible. However, in many applications, focusing on

average behavior such as the mean, variance, and autocorrelation, which can be computed from the probabilistic description or estimated from specific realizations, can provide useful results and insights. In particular, often Fourier transforms of random signals of interest do not exist; however, the autocorrelation functions usually have Fourier transforms and are amenable to further analysis. In particular, consider the system in Figure 3.1, where now the input is a realization of a random process. The output is therefore also a realization of a random process. Assuming the random process associated with the input is wide-sense stationary (i.e., it has a constant mean and an autocorrelation function that depends only on relative timing), with mean m_x and autocorrelation function $R_{xx}[m] = E\{x[n]x[n+m]\}$, the mean, cross-correlation, and autocorrelation of the output can be obtained as follows:

$$m_y = m_x \sum_{k=-\infty}^{\infty} h[k] = m_x H(e^{j0}) \quad 3.12$$

$$R_{yx}[m] = \sum_{k=-\infty}^{\infty} R_{xx}[m-k]h[k] \quad 3.13$$

$$R_{yy}[m] = \sum_{k=-\infty}^{\infty} R_{xx}[m-k]R_{hh}[k] \quad 3.14$$

where m_y and $R_{yy}[m]$ are the mean and the autocorrelation function of the output signal respectively. $R_{yx}[m] \equiv E\{y[n]x[n+m]\}$ is the cross-correlation between the output and input, and $R_{hh}[k] \equiv \sum_{l=-\infty}^{\infty} h[l]h[k+l]$ is the deterministic autocorrelation of $h[n]$. Note that the output is also wide-sense stationary. Similar expressions hold for continuous-time signals.

The power spectrum or power spectral density (PSD) of a signal is defined as the Fourier transform of the autocorrelation function. Since the autocorrelation function of the output is equal to the convolution of the autocorrelation function of the input with the deterministic autocorrelation of the system's impulse response, the power spectrum of the output is equal to the product of the power spectrum of the input with the power spectrum of the system (the latter corresponds to the magnitude squared of the system function). The term "power spectral density" is used because the function describes the frequency distribution of power (or variance) in a time series. In practical applications, one needs to estimate the PSD or the autocorrelation function from samples of a random process. A number of methods exist that provide estimates of these functions. A class of methods consists of Fourier transforming windowed versions of the signal and averaging the transformed versions. This method is referred to as periodogram averaging. Another class of methods consists of first estimating the autocorrelation sequence and then taking the Fourier transform of the estimate; the DFT is usually used in implementations of this method. It is important to note that there is an implicit assumption here that all the random signals are ergodic which, loosely defined, refers to the fact that time averages converge to ensemble averages.

3.2 Signal Detection and Estimation

Signal detection and estimation refers to the area of signal processing that seeks to extract information through the processing of information-bearing signals. The basic problem can be formulated as follows: we are interested in a signal $x[n]$ that we do not have direct access to, in other words, we do not have direct observations of $x[n]$. Instead we can only observe a related signal $y[n]$ obtained by processing $x[n]$ through some unknown (or partially known) system S . The *estimation* problem then consists of designing a system H that takes as input $y[n]$ and provides as output an estimate of $x[n]$, which we denote $\hat{x}[n]$. A schematic of a canonical signal estimation problem is given in Figure 3.2. Solutions to estimation problems include Bayesian parameter estimation, maximum likelihood estimation, and deconvolution methods such as Wiener filtering. Most *detection* problems, on the other hand, can be formulated as M -ary hypothesis testing problems where, based on observations of $y[n]$, we wish to decide among M possible hypotheses, ' h_1 ', ..., ' h_M ' related to the signal $x[n]$ (a simple version of this could be the presence or absence of $x[n]$). The processing typically includes two steps, a filtering step, which consists of processing $y[n]$ through a system H , and a testing step, which consists of applying a threshold test to the processed signal. A schematic of a canonical signal detection problem is given in Figure 3.3. Solutions to detection problems include minimax hypothesis testing, Neyman-Pearson hypothesis testing, and match filtering. An introduction to signal detection and estimation techniques is provided in [21].

Detection and estimation techniques have been applied to many engineering areas, including communications, control, seismology, radio astronomy, and medical signal processing. Radar is another classical application that has motivated many of the advances in this area. Recently, these techniques, and more generally Fourier analysis, have been applied to genomic and proteomic data. Some of these applications have been reviewed in a series of excellent review articles [22–24]. The general methodology in this area, which essentially consists of formulating the problem in the language presented in Figure 3.2 or Figure 3.3, usually includes five

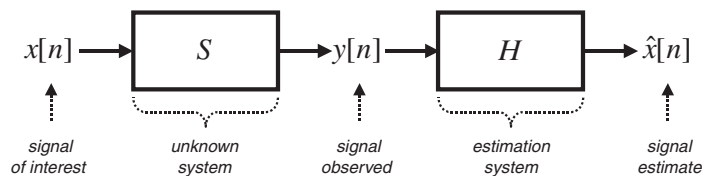


Figure 3.2 Canonical representation of a signal estimation problem.

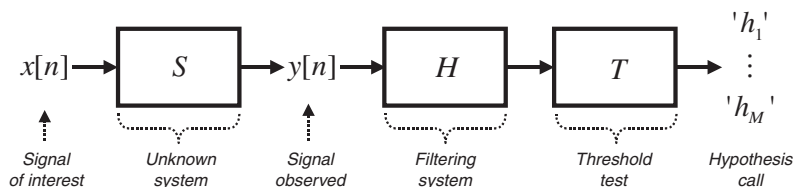


Figure 3.3 Canonical representation of a signal detection problem.

Table 3.1 General methodology for analyzing genomic and proteomic data.

-
- (1) Describe and identify the measurement system (S in Figure 3.2 and Figure 3.3).
 - (2) Define the signal of interest ($x[n]$ in Figure 3.2 and Figure 3.3). This step usually involves a mapping from the biological space into a numerical space.
 - (3) Formulate the signal processing problem (e.g., estimation, detection, or analysis). This step usually involves designing a system to process the measured signal and compute an output signal.
 - (4) Develop techniques to solve the problem and compute the output signal.
 - (5) Interpret the results in a biological context. This step usually involves reversing the mapping introduced in Step 2.
-

steps as outlined in Table 3.1. In the next sections, we present examples of biological questions posed in the framework of Table 3.1. We start with DNA sequencing and describe the basecalling problem; we then proceed to analyzing the DNA sequence and reveal interesting spectral features of DNA. The problem of gene identification is then formulated and discussed. We end this section with protein sequence analysis and describe signal processing methods for identifying protein hotspots.

3.2.1 DNA Sequencing

DNA sequencing, the process aimed at identifying the sequence of bases in a given DNA molecule, has undergone major advances over the past decade, primarily driven by the needs of the Human Genome Project [25]. The process itself contains three steps: (1) DNA sample preparation, (2) electrophoresis, and (3) processing. The first two steps are experimental and the third one is analytical. Processing the electropherogram data (the output of electrophoresis) in order to identify the DNA sequence of interest includes two main steps: a prefiltering step aimed at conditioning the signal and increasing the signal-to-noise ratio, and a basecalling step aimed at identifying the underlying DNA sequence. Prefiltering and basecalling involve a number of interesting challenges, some of which have been addressed using signal processing techniques.

3.2.1.1 Signal and System Definition for the DNA Sequencing Problem

In order to formulate DNA sequence identification as a signal processing problem, we first need to define the signal and system of interest. To identify the signal, we need to map the DNA sequence, which is composed of character strings corresponding to the nucleic acids, into a numerical signal that is amenable to analysis using signal processing techniques. This corresponds to Step 2 in the general methodology outlined in Table 3.1. A simple mapping corresponds to defining four binary indicator sequences, $x_a[n]$, $x_t[n]$, $x_c[n]$, and $x_g[n]$, corresponding to each one of the four nucleotides, A, T, C, and G respectively, which indicate the presence or absence of each nucleotide at the n^{th} base-pair position [26]. For a DNA sequence with N nucleotides, the sequences are defined for $0 \leq n \leq N - 1$. Note that “time” here (i.e., the independent variable n) corresponds to base-pair position. An example of such mapping is shown in Figure 3.4.

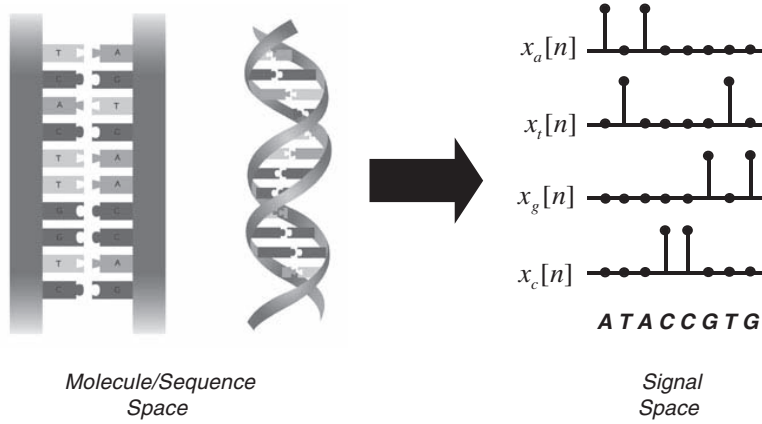


Figure 3.4 Example of DNA sequence mapping into a collection of four indicator sequences.

Once the signal has been defined, the system under consideration needs to be formulated. Specifically, DNA sample preparation and electrophoresis can be viewed as a system that takes a DNA signal as input (the four binary indicator sequences corresponding to the four base pairs) and outputs a distorted version of these signals, which is the measured signal as shown in Figure 3.5 (for clarity, we display the envelope of the output signal in the figure). The DNA sequence identification problem can then be formulated as estimating the signals $x_a[n]$, $x_t[n]$, $x_c[n]$, and $x_g[n]$ from the measured signals $\tilde{x}_a[n]$, $\tilde{x}_t[n]$, $\tilde{x}_c[n]$, and $\tilde{x}_g[n]$. We next present three approaches to finding solutions to this problem.

3.2.1.2 DNA Sequence Estimation using Wiener Filtering

A simple approach to addressing the DNA sequence estimation problem is to use LTI filtering for estimating $x[n]$ as shown in Figure 3.6 (for simplicity we use $x[n]$ to refer to any one of the four DNA signals: $x_a[n]$, $x_t[n]$, $x_c[n]$, or $x_g[n]$). Specifically, we want to determine the impulse response or frequency response of the LTI

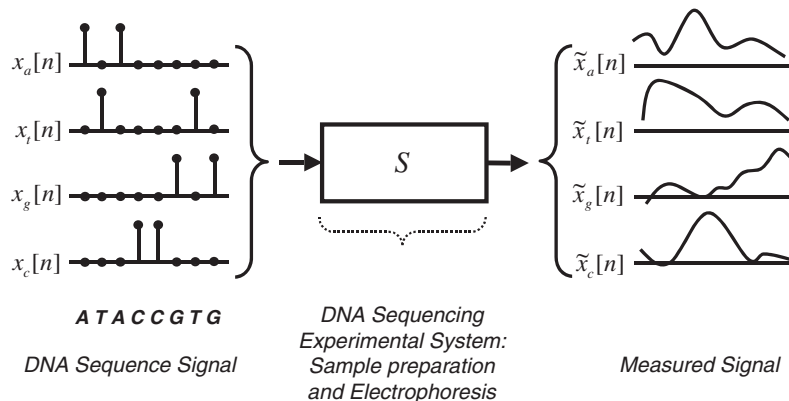


Figure 3.5 Signal processing model of DNA sequencing. (For the output signal, we display the envelope of the signal for illustration purposes.)

system, $h[n]$, in Figure 3.6 such that the filter output $\hat{x}[n]$ is the minimum mean-squared-error (MMSE) estimate of the target process $x[n]$, that is:

$$\min_{h[\cdot]} (\varepsilon = E\{e^2[n]\})$$

where $e[n] \equiv \hat{x}[n] - x[n]$

3.15

Assuming $x[n]$ is a wide-sense stationary random process where the particular DNA sequence is a realization of this process and assuming that the measurement process, $\tilde{x}[n]$, is jointly wide-sense stationary with $x[n]$, the resulting filter, $h[n]$, corresponds to a Wiener filter. Specifically, it can be shown that for the optimal system, the cross-correlation between the input and output of the estimator equals the cross-correlation between the input and target output [27]. Equivalently, the filter must satisfy the following equation:

$$\sum_k h[k] R_{\tilde{x}\tilde{x}}[m-k] = R_{\tilde{x}x}[m]$$
3.16

Equation (3.16) represents a set of linear equations that need to be solved for the impulse response values. If the filter is restricted to be length N , then there are N equations in the N unrestricted values of $h[n]$ and the problem can be easily solved using existing efficient methods. If the filter is not restricted in length, then taking the Fourier transform of (3.16) and solving for the filter system function gives the following solution:

$$H(e^{j\omega}) = \frac{S_{\tilde{x}x}(e^{j\omega})}{S_{\tilde{x}\tilde{x}}(e^{j\omega})}$$
3.17

A number of different techniques exist to implement the system function in (3.17); the reader is referred to [8] for examples of such techniques.

Applying Wiener filtering to separate the signal peaks from the system blur caused by electrophoresis is a good estimation method in the absence of spatial variance (i.e., diffusion effects) since system blur is a linear process. However, in practice, diffusion effects, which can be significant, introduce spatial variance that makes estimation using LTI filtering prone to errors. A number of nonlinear techniques have therefore been developed to deal with diffusion effects and are being used to detect DNA sequences. We next provide two examples of such techniques.

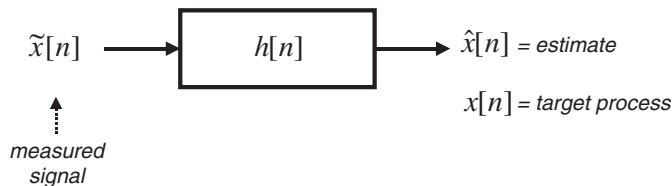


Figure 3.6 DNA sequence estimation using LTI filtering.

3.2.1.3 Homomorphic Blind Deconvolution Applied to DNA Sequencing

The effect of electrophoresis can be represented as the convolution of a blurring function with the DNA sequence signal followed by additive noise distributed throughout the image. The blurring function is caused by the emission pattern of the fluorescent labels and the diffusion width of the bands in the gel. Estimating the DNA sequence therefore consists of deconvolving the measured signal from the blurring function to recover the DNA sequence signal. As mentioned earlier, this process is linear in the absence of diffusion effects and therefore linear filtering methods such as Wiener filtering lead to good results in this case. However, in practical situations where diffusion effects cannot be ignored, Wiener filtering can lead to significant errors. In this case, deconvolution based on homomorphic blind deconvolution leads to better results [28].

Homomorphic signal processing deals with a class of nonlinear systems that obey a generalization principle of superposition. Specifically, these systems are represented by applying algebraic linear transformations between the input and output spaces. Homomorphic systems therefore convert nonlinear inputs to signals suitable for linear processing. For example, multiplicative systems are converted to additive systems suitable for linear processing by computing the logarithms of the two multiplied signals. Detailed discussions of homomorphic deconvolution is provided in [29]. Ives et al. [28] applied homomorphic processing in combination with blind deconvolution to the problem of DNA sequencing. Blind deconvolution is needed since the two convolved signals (system blur and DNA sequence) are both unknown beyond a general description. Figure 3.7 illustrates the processing steps involved in homomorphic deconvolution. Briefly, the measured signal is the convolution of the DNA signal we are trying to estimate with a blurring function. The spectrum of the measured signal is therefore the product of the Fourier transforms of the original DNA signal and the blurring function. Noise due to diffusion effects prevents processing the spectrum of the measured signal by simple division by the Fourier transform of the blurring function in order to recover the original signal. Instead, in homomorphic processing, the complex logarithm of the spectrum is computed and therefore the product of the Fourier transforms is converted into a sum of log-spectra. The blurring function can be modeled as a Lorentzian point spread function whose spectrum is a straight line with negative slope. The slope is dependent on the width of the electrophoresis bands. Most of the energy in straight lines is at low frequencies, in contrast to the widely scattered frequency distribution of the DNA signal peaks. Applying a generalized high-pass filter should therefore considerably attenuate the blurring function while preserving most of the DNA signal. This step corresponds to the blind deconvolution since the exact blurring function is not known. Since the Lorentzian point spread function is real and even, its spectrum and log-spectrum are real, and therefore processing only the real part of the spectrum (or log-spectrum) of the measured signal is needed since the imaginary part is due entirely to the DNA signal we seek to estimate. Hence, only the real part of the log-spectrum undergoes further processing in Figure 3.7. The inverse Fourier transform of the real part of the log-spectrum ($\tilde{C}$ in Figure 3.7) is referred to as the cepstrum (variant of “spectrum” to indicate that a second Fourier transform was computed). As discussed above, the cepstrum of the blurring function is large at low

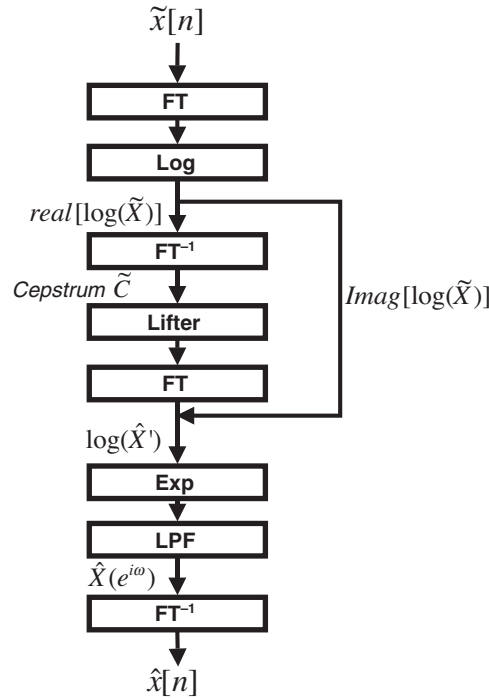


Figure 3.7 Processing steps in homomorphic deconvolution. FT and FT^{-1} correspond to the Fourier transform and inverse Fourier transform respectively. Log and Exp correspond to the logarithm and exponential operations respectively. LPF corresponds to a low-pass filter.

quefrequencies (variant of “frequency”) and therefore a high-pass lifter (variant of “filter”) is multiplied by the cepstrum to reduce the effect of the blurring function. The Fourier transform of the output of the lifter is then added to the imaginary part of the log-spectrum of the measured signal to recover the processed log-spectrum. The processed spectrum, $\hat{X}'(e^{j\omega})$, is then recovered and filtered with a low-pass filter to remove additional high-frequency noise. Taking the inverse Fourier transform of the output of the low-pass filter leads to the estimated DNA signal $\hat{x}[n]$.

The algorithm was applied on a digitized electropherogram containing 566 bands. Figure 3.8 shows example plots of the result. It was found that the algorithm had an error rate of 1.06% where, for the first 400 bases, no errors were made. This was significantly better than reports from fluorescence-base sequencing instruments developed at the same time the paper was published as well as commercial film-based readers, especially beyond 300 nucleotides.

A number of alternative deconvolution methods have been subsequently proposed. In particular, Berno [30] proposes a nonlinear filter to deconvolve the data that operates on the second and fourth derivatives of the data. The differentiation component of the filter are implemented in the Fourier domain with a high cutoff component to dampen any resulting noise. A nonlinear reconstruction algorithm using iterative deconvolution is proposed by Zhang and Allison [31]. A contract mapping function is designed and shown to provide improvements over methods that use linear Wiener filtering.

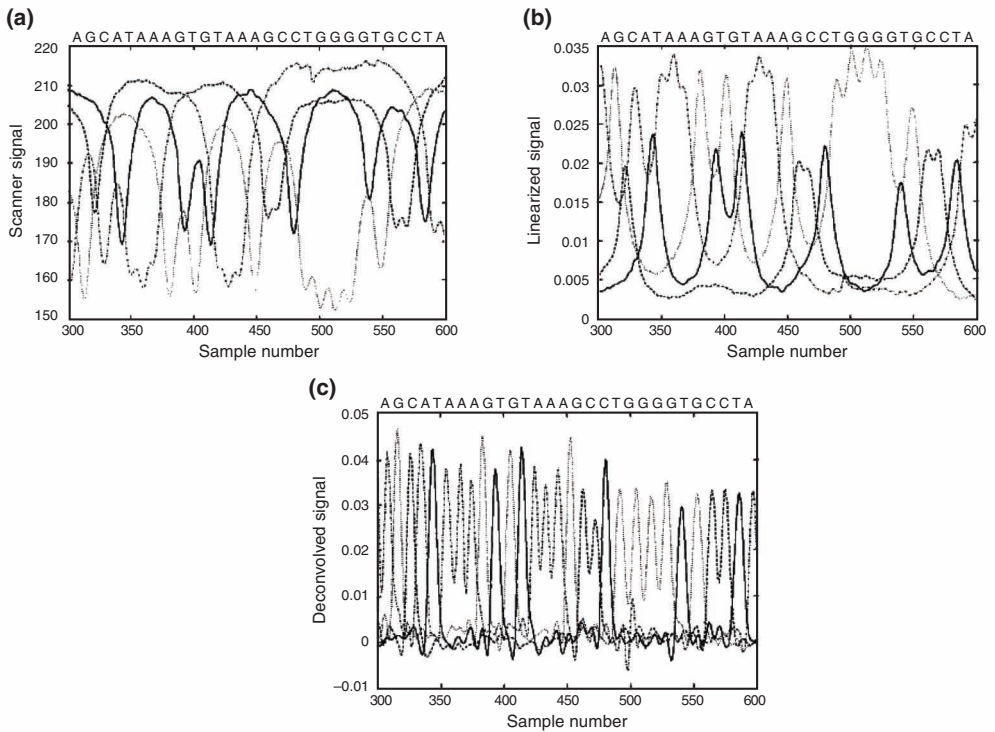


Figure 3.8 Example plots of data prior to and through homomorphic blind deconvolution. (a) scanned sequencing lanes, (b) linearized data, (c) deconvolved and aligned lanes. (From: [28].) © 1994 IEEE. Reprinted with permission.

3.2.1.4 Model-Based Estimation Techniques

The homomorphic deconvolution approach discussed above models the effects of electrophoresis as a convolution of a blurring function followed by additive noise. More generally, sequencing effects can be decomposed into a cascade of distortions representing the different processes involved in the experimental system. Specifically, there are four main distortions introduced by sequencing: (1) loading artifacts, (2) diffusion effects, (3) fluorescence interference, and (4) additive instrument noise [32]. System S in Figure 3.5 can therefore be decomposed into the four systems shown in Figure 3.9.

In the context of Figure 3.9, processing the electropherogram signal consists of undoing each distortion step. Specifically, first *denoising* aims at removing experimental noise introduced by the gel and electronics as well as by the optical equipment. The noise is usually modeled as an additive white Gaussian process and the denoising filter is typically a low-pass filter since DNA fluorescence is a low-frequency signal. *Color separation* is then carried out to remove cross-talk between the four channels due to fluorescence interference. The distortion is usually modeled as a linear mixing of the four signals. Specifically, let $\tilde{x}$ be the original desired four-dimensional signal and M denote the 4×4 mixing matrix, then the measured signal is $\tilde{x}_m = M\tilde{x}$; the original signal can therefore be recovered by inverting the mixing matrix. M is not always known and therefore usually needs to be estimated. After recovering the original color signal, *baseline correction* is carried out to remove a

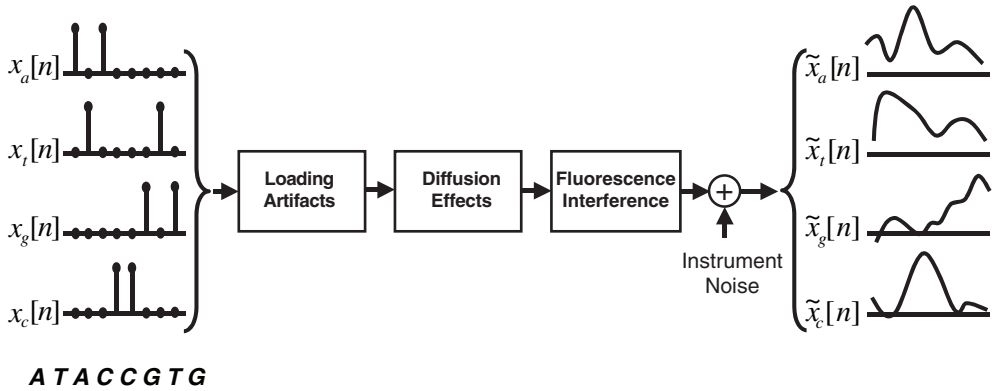


Figure 3.9 Detailed signal processing model of DNA sequencing.

DC (zero frequency) offset created by background fluorescence due to experimental conditions. Note that the value of the DC offset may not be constant during the experiment and therefore adaptive techniques need to be used. The baseline correction filter is effectively a high-pass filter. *Mobility shift correction* is then needed to undo the time warping due to the effect of the fluorescent tags on the mobility of the DNA molecules. The four fluorophores used for labeling the DNA fragments typically have different electrophoretic mobilities. In addition, this step corrects for the nonuniform peak spacing generated by the loading artifacts. The output of this final step leads to a signal with uniform peak spacing that is ready for *basecalling*.

A number of approaches have been proposed to refine the models of each step in Figure 3.9. For example, in [33], noise filtering includes two steps. The first step deals with background noise, detector noise, and other noise from the operating environment, which typically have a white spectrum, while the second step deals with low-frequency noise generated by changes in the background light level during collection. These variations may be caused by deformation of the gel due to heating, the formation of bubbles in the path of the laser, or variations in laser output power. As a result, a band-pass filter is designed to remove both high-frequency noise and low-frequency background variation. The filter has a very sharp low-frequency cutoff in conjunction with a more gradual high-frequency cutoff. Mobility shift corrections are compensated using a small constant offset in one or more channels to produce the correct alignment, since dye-mobility shift is nearly linear over large regions. Two methods to estimate the color filter matrix M are proposed in [34]. The first method guarantees the nonnegativity of the estimate; however, it suffers from a slow convergence rate. The second method runs much faster; however, the nonnegativity of the estimate is not guaranteed. Li in [35] uses simulated data generated based on the DNA sequencing model as a training set to develop and optimize basecalling methods. Additional techniques recently used include Bayesian probabilistic frameworks [36], an improvement to the Bayesian framework that allows removal of slowly varying background noise and that is able to track nonstationarity in the various processes [37], hidden Markov models [38], and graphical models [39].

Determining the impact that these methods have had on DNA sequencing algorithms implemented in widely used commercial software is difficult since full algorithmic details are not typically disclosed for these programs. However, it is believed that most programs use variants of the methods discussed above. Currently, the most widely used basecaller is an open-source program called Phred [40, 41]. The software is heavily optimized for slab-gel sequencers. It uses a four-phase procedure to determine the sequence from a processed signal, which may be obtained from other commercial software such as the ABI analysis software. The first phase consists of finding the idealized location of the base peaks using Fourier analysis, starting in regions of the electropherogram that have the most uniform spacing. Observed peaks are then identified in the second phase and matched to predicted peak locations in the third phase, omitting some peaks and splitting others. Finally, in the fourth phase, the uncalled observed peaks are checked, leading to additional insertions if needed. While Phred usually performs well, leading to low-error reads, its performance degrades when spacing between peaks changes abruptly along the traces, which usually happens in MegaBACE sequencers. In this case, the LifeTrace algorithm has been shown to perform better [42]. The cross-correlation of the measured signal with an ideal Gaussian-shaped peak is computed to determine the ideal peak location as a first step. Two iterations of quality filtering are then carried out to enhance the signal. Quality scores are finally computed to allow assessment of the reliability of the call, discriminating high-quality from low-quality calls. The reader is referred to the original papers for more details.

3.2.2 Gene Identification

Once the DNA sequence has been identified, it needs to be analyzed to identify genes and coding sequences. In this section, we describe some of the signal processing techniques that have been developed to identify genes. We start by discussing some of the underlying properties of DNA signals, including their spectral properties, and then present signal processing methods for gene prediction.

3.2.2.1 DNA Signal Properties

As shown later in this section, understanding the properties of the DNA signal is useful in identifying coding genes. Consider the autocorrelation function of the indicator sequence $x_a[n]$ of adenosine: $R_{x_a x_a}[m] = \sum_k x_a[k]x_a[k+m]$ (similar expressions hold for indicator sequences of the other nucleotides). Taking the Fourier transform of the autocorrelation function gives the power spectrum $S_{x_a x_a}(e^{j\omega})$. One should be reminded that since we are dealing with finite sequences, the lowest meaningful frequency is $\omega_0 = 2\pi/N$. Figure 3.10 shows $S_{x_a x_a}(e^{j\omega})$ for the first one million bases of the genome of the bacterium *Aquifex aeolicus* [23]. The nonflat shape of the spectrum reveals correlations at low frequencies, indicating that base pairs that are far away seem to be correlated in a statistical sense. These long-range correlations of nucleotide sequences were first reported in 1992 by Peng et al. [43] and by Voss [26]. Through systematically examining the power spectrum of a number of organisms, Voss demonstrated the power-law (also referred to as $1/f$, where f is

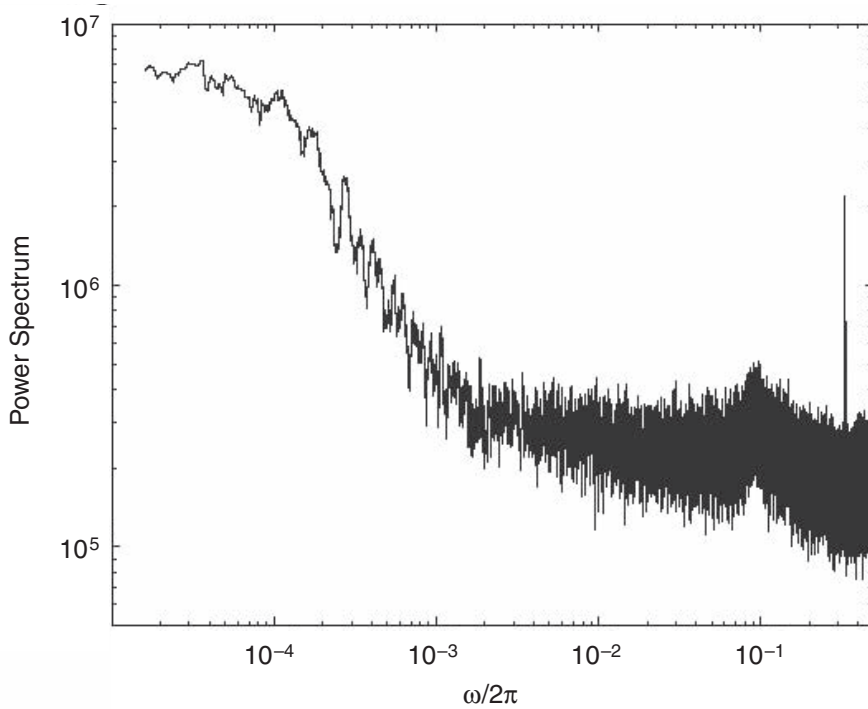


Figure 3.10 Power spectrum of the first one million bases of the genome of the bacterium *Aquifex aeolicus*. (From: [23].) © 2004 *The Journal of the Franklin Institute*. Reprinted with permission from Elsevier.

frequency) property of DNA sequences. In particular, he showed that $S_{x_0, x_0}(e^{j\omega})$ is proportional to $1/\omega^\beta$, where the value of β varied across bases and across organisms. Vieira [44] later studied the statistical properties of DNA chains of the complete genomes of 13 microbial species and showed, using periodogram averaging, that the power spectrum flattens in the low-frequency limit. A good review on the power-law properties of DNA sequences is provided by Li in [45], and more recent results on the human genome are reported in [46]. It has also been suggested that power-law behavior in natural systems can be explained by a duplication-mutation model [47].

A localized increase in power at $2\pi/3$ is also apparent in Figure 3.10, as shown by the thin peak at $2\pi/3$. This increased correlation corresponds to the tendency of nucleotides to be repeated along the DNA sequence with period 3 and is indicative of coding regions. In particular, early speculations attributed this effect to the triplet nature of the codon and potentially codon bias (unequal usage of codons) as well as the biased usage of nucleotide triples in genomic DNA (triplet bias), which is usually specific to a given organism. However, Tiwari et al. [48] and more recently Yin and Yau [49] have shown that the period-3 property is not affected by codon bias. Yin and Yau also showed that the amino acid composition, not the ordering of the amino acids, in proteins determines the period-3 property. The period-3 property of coding regions seems to be generated by the unbalanced nucleotide distributions in the three codon positions. As we show next, the period-3 property of coding regions can be exploited to develop methods to identify genes within DNA sequences [50].

3.2.2.2 DNA Signal Processing for Gene Identification

As sequences become more readily available through a multitude of genome projects, developing computational tools to automatically identify coding regions is becoming a necessity. The gene identification problem has been defined by Fickett [51] as “the problem of interpreting nucleotide sequences by computer, in order to provide tentative annotation on the location, structure, and functional class of protein-coding genes.” More practically, the success of gene identification algorithms is measured in terms of their ability to correctly predict the amino acid sequence of protein products and potentially provide some insight into their function. A number of methods have been developed to address this problem. The premise of all these methods is to exploit the period-3 property of coding regions by processing the DNA signal to identify regions with strong period-3 correlations.

As noted earlier the period-3 property of a DNA sequence implies that the Fourier spectrum of the indicator sequences is large at $2\pi/3$ or equivalently that the DFT coefficients corresponding to $k = N/3$ are large. In order to process the DNA signal, we first define $X_a[k]$, $X_t[k]$, $X_c[k]$, and $X_g[k]$ to be the N -point DFT of each indicator sequence, where

$$X_i[k] = \sum_{n=0}^{N-1} x_i[n] e^{-j2\pi kn/N}, \quad 0 \leq k \leq N-1 \quad 3.18$$

for $i = a, t, c, g$. The DNA spectrum, $S_x[k]$, is then defined as the sum of the spectra of the indicator sequences:

$$S_x[k] = |X_a[k]|^2 + |X_t[k]|^2 + |X_c[k]|^2 + |X_g[k]|^2 \quad 3.19$$

Let $P_x(N/3)$ be the signal-to-noise ratio at $k = N/3$ in the spectrum, that is,

$$P_x(N/3) = \frac{S_x[N/3]}{\overline{S_x}}$$

where $\overline{S_x} \equiv \frac{1}{N} \sum_{k=0}^{N-1} S_x[k]$ is the average power. Tiwari et al. [48] observed that for most coding sequences in a variety of organisms, $P_x(N/3)$ is large, while non coding sequences have a low $P_x(N/3)$. Figure 3.11 shows examples of typical Fourier spectra for the coding and noncoding regions from *S. cerevisiae* chromosome III.

A simple method to potentially identify coding sequences therefore consists of computing $P_x(N/3)$ for different regions of the DNA sequence and identifying the ones that have a large value at this frequency. Note that since we are only interested in a single point in the DFT, there is no need to compute the entire spectrum, therefore saving substantial computation. It should be noted, however, that due to the nature of the windowing operation, there are tradeoffs between time-domain resolution (base-pairs resolution) and frequency-domain resolution (ability to resolve the peak over background ($1/f$) noise). A number of alternative methods to windowing have been proposed [22, 23, 52–54]. In particular, [54] and [22] explore broader mappings to define the DNA signal that enhance the ability of appropriate filters to identify coding regions. In [23, 52, 53] Vaidyanathan and Yoon propose

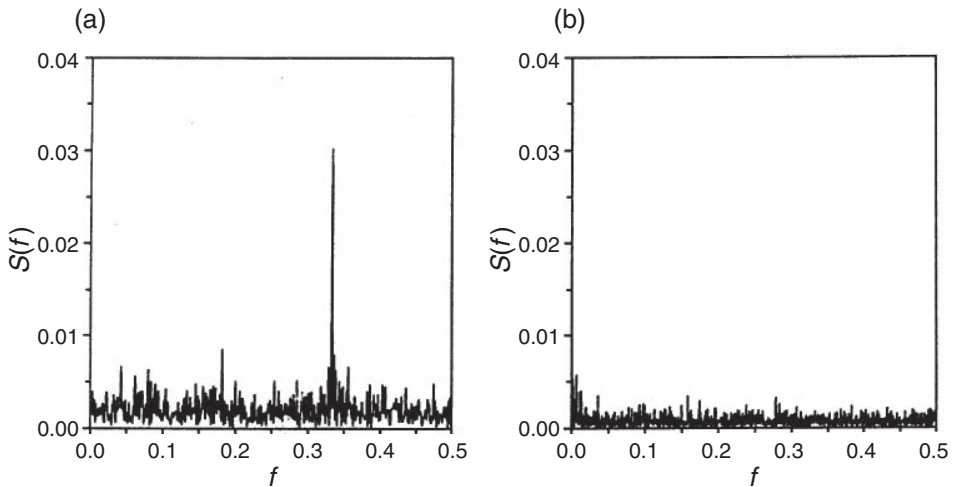


Figure 3.11 Typical Fourier spectra for (a) a coding stretch of DNA and (b) a noncoding stretch from *S. cerevisiae* chromosome III. (From: [48].) © 1997 Oxford University Press. Reprinted with permission.

methods that consist of processing the indicator sequences through a bandpass filter localized around $\omega_0 = 2\pi/3$. The output of this filter is expected to be large in coding regions due to the period-3 property and almost zero in noncoding regions. The same tradeoff between frequency and time-domain resolution as the windowing method holds for the band-pass filter. Specifically, the length of the filter impulse response in the time domain corresponds to the window length in the frequency domain. Vaidyanathan and Yoon describe two different classes of filters in [52]: infinite impulse response (IIR) antinotch filters and multistage filters. The multistage filters provide better stop-band attenuation at the expense of additional computation. Figure 3.12 compares the performance of the three filtering methods (window, IIR antinotch, and multistage filters) in predicting the five exons for gene F56F11.4 in *C. elegans* chromosome III. More recently, a technique using a single digital filter operation followed by a quadratic window operation has been proposed [55]. The technique was applied to gene F56F11.4 in *C. elegans* chromosome III and was shown to suppress nearly all the noncoding region, therefore improving the likelihood of correctly identifying coding regions.

DNA sequence analysis can be performed beyond gene identification to identify other DNA regions of interest. For example, CpG islands, which are regions of the DNA rich in the dinucleotide CpG, have been shown to correspond to gene markers as they tend to be located upstream of transcription factors binding regions for many genes. A method for identifying CpG islands using a bank of IIR low-pass filters has been suggested by Yoon and Vaidyanathan in [56]. The reader is referred to the original paper for more detail.

The techniques presented here are unable to reliably locate coding regions in sequences that do not have period-3 property. In this case, methods using hidden Markov models perform better [57, 58]. In fact, most of the commercial gene-finding software such as Fgenesh and Genescan are based on hidden Markov models. Most programs also include algorithms that use homology mapping to known proteins. These programs, however, do not provide sufficient support for gene an-

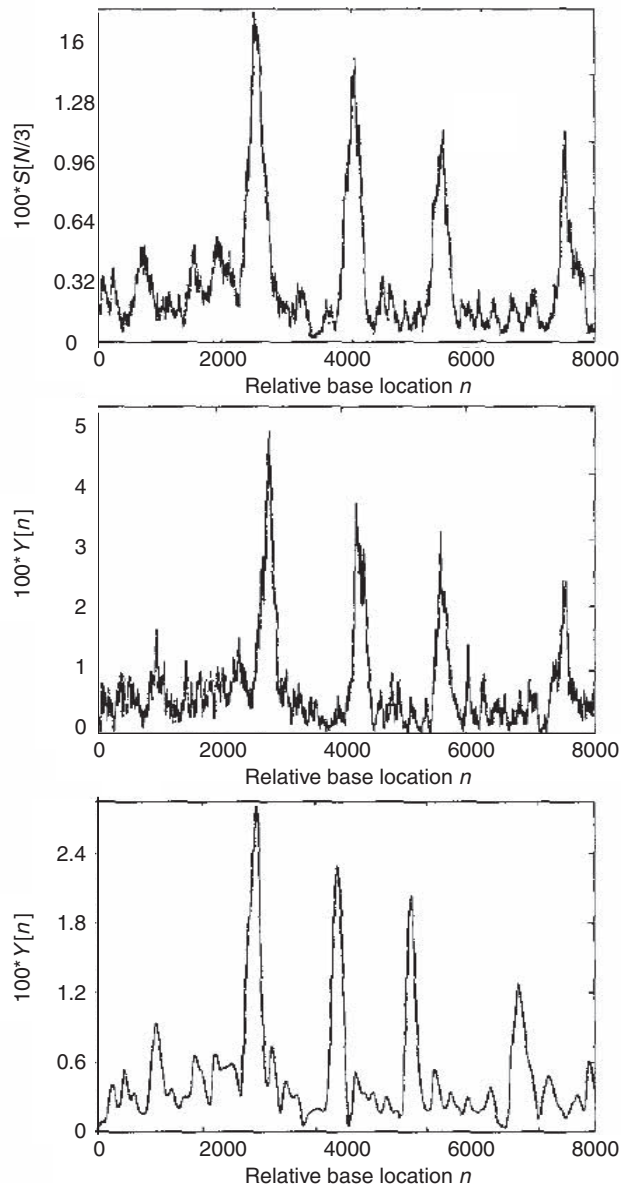


Figure 3.12 Comparison of gene identification methods for gene F56F11.4 in the *C. elegans* chromosome III. Top plot: the DFT-based spectrum. Middle plot: the antinotch filter output. Bottom plot: the multistage narrow-band band-pass filter output. (From: [52].) © 2002 IEEE. Reprinted with permission.

notation when dealing with novel genomes. An algorithm for gene identification in novel eukaryotic genomes was recently published. Details of the approach are described in [59].

3.2.3 Protein Hotspots Identification

Once coding regions have been identified, the corresponding protein sequence can be determined by mapping the coding region to the amino acid sequence using the genetic code. The problem then consists of determining the function of the proteins

identified using the information provided by the amino acid sequence. This problem has been posed in many forums and addressed by different scientific communities including biologists, mathematicians, physicists, and engineers. It takes on different forms depending on the community and the methodology used. For example, the well-known protein folding problem whose goal is to predict the three-dimensional structure (folding) of a protein from the amino acid sequence focuses on the premise that it is the three-dimensional structure of proteins that leads to their functionality and therefore predicting the structure would provide insight into protein function. This is a very important problem and remains one of the most basic unsolved problems in computational biology. We do not address the protein folding problem here. Instead, we focus on a simpler problem that consists of identifying common functionality among a collection of proteins. Specifically, we are interested in the following question: can we identify common functionality (hot spots) of a given protein sequence with other proteins using signal processing techniques? This problem has also been explored by sequence alignment techniques; here, however, we focus on signal processing methods.

3.2.3.1 Protein Signal Definition

As is the case for nucleotide sequences, the first step to address the problem of interest consists of mapping amino acid sequences into numerical sequences amenable to analysis. Cosic [60] defined a mapping based on the resonant recognition model (RRM), which uses a physical parameter value that is relevant to the biological activity of the protein. Specifically, each amino acid is represented by the value of the electron-ion interaction potential (EIIP), which corresponds to the average energy states of all valence electrons. The EIIP values for the 20 amino acids are given in Table 3.2. An N length amino acid sequence can therefore be mapped into a discrete-time signal, $p[n]$, where the independent variable n corresponds to the amino acid location along the protein sequence and the value of $p[n]$ corresponds to the EIIP value of the amino acid at location n . Analyzing this sequence provides insight into the functional properties of the underlying protein.

3.2.3.2 Cross-Spectral Properties of Proteins with Common Function

The spectral content of protein signals as defined above is complex and very hard to interpret in isolation, that is, it is very hard to use spectral information to infer functional properties about the underlying protein. However, extracting the common spectral characteristics of multiple sequences sharing similar function leads to

Table 3.2 The electron ion interaction potential (EIIP) values for amino acids.

<i>Amino Acid</i>	<i>EEIP</i>	<i>Amino Acid</i>	<i>EEIP</i>	<i>Amino Acid</i>	<i>EEIP</i>	<i>Amino Acid</i>	<i>EEIP</i>
Leu	0.0000	Glu	0.0058	Tyr	0.0516	Cys	0.0829
Ile	0.0000	Pro	0.0198	Trp	0.0548	Thr	0.0941
Asn	0.0036	His	0.0242	Gln	0.0761	Phe	0.0946
Gly	0.0050	Lys	0.0371	Met	0.0823	Arg	0.0959
Val	0.0057	Ala	0.0373	Ser	0.0829	Asp	0.1263

insight into the nature of their shared biological functions, as demonstrated by Cosic [60]. Specifically, let $p_1[n], \dots, p_M[n]$ correspond to the numerical sequences of M proteins sharing a common function and denote $P_1[k], \dots, P_M[k]$ the corresponding N point DFT sequences. The cross-spectral function is defined as $M[k] \equiv |P_1[k]| \times |P_2[k]| \times \dots \times |P_M[k]|$.

Empirical studies of over 1,000 proteins from 25 functional groups have shown that the presence of a peak frequency in $M[k]$ with a significant signal-to-noise ratio (implying that all of the sequences $P_i[k]$ have one frequency component in common) is related to biological function as long as the following three criteria are met:

1. The cross-spectral function has only one peak frequency.
2. Biologically unrelated sequences do not exhibit a significant peak.
3. Different biological functions exhibit different peak frequencies.

Characteristic frequencies for 28 functional groups of proteins including kinases and oncogenes are given in [60]. A possible interpretation of the existence of characteristic frequencies is the manifestation of resonant recognition between macromolecules at a distance. Knowledge of these characteristic frequencies can be exploited to predict protein hotspots.

3.2.3.3 Prediction of Protein Hotspots

A protein hotspot is a region in the amino acid sequence of the protein that corresponds to a minimal functional domain, that is, an active site such as a binding area in the three-dimensional structure of the protein. Characteristic frequencies can be used to predict protein hotspots. One of the earlier methods consisted of using inverse Fourier transforms to identify the collection of amino acids that are most sensitive to changes in the characteristic frequency [60] using the following three steps:

1. Determine the characteristic frequency for the biological function of interest by analyzing cross-spectra of a group of proteins with the corresponding biological function.
2. Change the amplitude of the characteristic frequency in the spectrum of the protein of interest until a minimum number of “hotspot” amino acids that are least sensitive to further changes in the amplitude of the characteristic frequency is reached.
3. Derive a numerical sequence from the modified spectrum using inverse Fourier transforms.

This inverse Fourier transform method allows the identification of a specific number of single amino acids that contribute to a particular frequency. The protein active sites, however, usually correspond to a domain within the protein sequence rather than a collection of distributed amino acids. Methods allowing the identification of protein domains have been developed using time-frequency analysis. In particular, a short-time discrete Fourier transform method is proposed in [61]. In this method, the short-time DFT of the protein signal is first computed and its columns are then multiplied by the DFT coefficients. Figure 3.13 shows an exam-

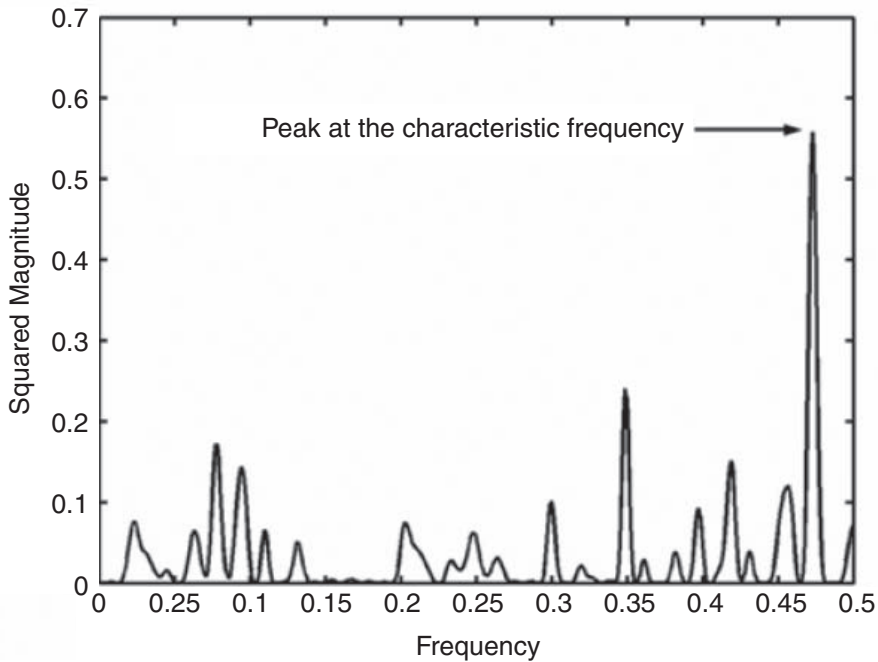


Figure 3.13 Consensus spectrum of cytochrome C proteins. The peak corresponds to the characteristic frequency. (From: [61].) © 2004 IEEE. Reprinted with permission.

ple of this methodology applied to cytochrome C proteins. Wavelet transforms have also been used to resolve protein domains; details on these methods can be found in [62, 63]. This kind of analysis can also be used for peptide engineering where peptides (amino acid sequences) with a given function can be designed using corresponding spectral characteristics. The reader is referred to [60] for more detail.

3.3 System Identification and Analysis

So far, the discussion has focused on signals. In particular, we provided examples on how DNA and protein signals can be estimated and detected using signal processing techniques. Once these signals have been defined and identified, the next step consists of understanding how they get processed by living systems. In particular, cells are constantly exposed to a multitude of signals coming from their environment such as hormones, growth factors, and signals from other cells. These signals are sensed by receptors at the cell surface that act as the cell's antennas. Receptors transduce the signals into the cell, through conformational changes, initiating a chain of signaling events leading to changes in gene expression and eventually to some kind of response such as cell death or cell proliferation. This process can be summarized as follows: a cue, or equivalently an input, triggers a receptor or antenna, which then starts a signaling pathway or signal processing/computation engine leading to some kind of response or output as shown in Figure 3.14.

If we consider multiple signaling pathways, ideally different inputs or ligands would trigger different antennas or receptors, which would then initiate different

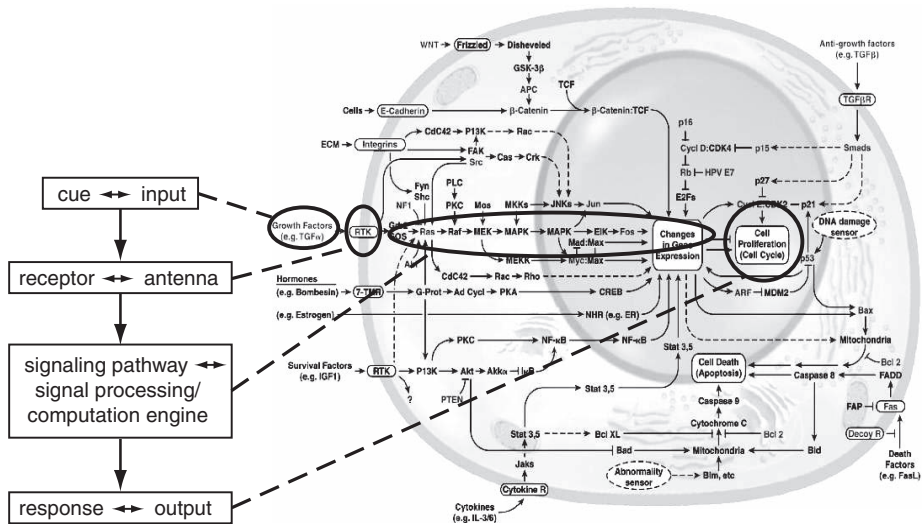


Figure 3.14 Signal processing view of the cell. (Cell cartoon adapted from [64a]. © 2000 Elsevier. Reprinted with permission.)

signaling cascades and eventually lead to distinct, clear outputs. However, things are not that simple in biology. In particular, ligands can trigger multiple receptors and some molecules or proteins are shared by different signaling pathways, leading to what is traditionally referred to as cross-talk or interference. This interference is one of the causes of drugs side effects whereby a drug initially designed to trigger a given signaling pathway simultaneously affects a second signaling pathway. Recently, our view of intracellular signaling has changed and cross-talk is no longer necessarily perceived as an interference or nuisance but rather as an essential component of signaling cascades enabling effective coordination among multiple signaling pathways. The traditional molecular view of signaling is therefore evolving to a systems level view where instead of considering isolated input/output pairs, multiple inputs and outputs are studied in concert. Specifically, a collection of inputs triggers a set of receptors that then triggers a signaling *network* rather than linear signaling cascades leading to a coordinated response as shown in Figure 3.15. In other words, the input and output are not scalar functions but multidimensional functions linked by systems defined by signaling networks. Our goal is to identify, characterize, and analyze the systems defining the input/output relationships implemented by signaling networks. The focus in this section is therefore on the signaling networks rather than the individual DNA and protein signals studied in the last section. Examples of signal processing systems within cells include energy production systems (metabolism), RNA transcription systems, systems governing the cytoskeleton, gene regulation systems, and protein signaling systems. Here, we focus on gene regulation and protein signaling as examples of such systems.

The recent emergence of high-throughput technologies for molecular biology is making the systematic study of gene and protein signaling networks a possibility. The development and widespread use of gene expression and protein arrays is generating massive amounts of data that need to be analyzed and classified. As a result, there has been an increased need for sophisticated mathematical tools for data mining and understanding. A number of modeling frameworks have been generated

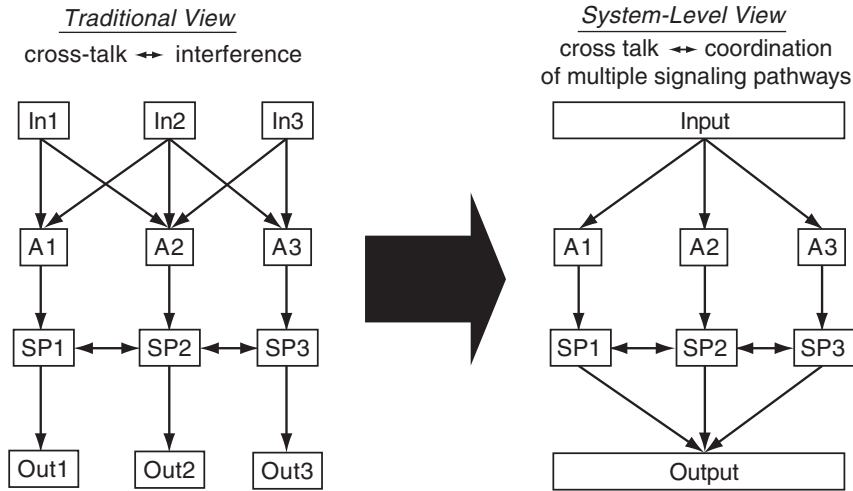


Figure 3.15 Signal coordination at the cell level. Evolution of the traditional molecular view of signaling into a system level view. In, A, SP, Out correspond to the different inputs, antennas (receptors), signaling pathways, and outputs respectively.

over the past few years, and a number of reviews provide good comprehensive highlights of the different modeling approaches and philosophies [64–66]. In the next sections, we present selected examples highlighting some of the signal processing approaches to modeling gene regulation and protein signaling. We focus on two classes of modeling techniques: a nonparametric method that allows the identification and analysis of the underlying system without prior assumptions about the relationships and interdependencies among the system components, and a model-based method allowing the incorporation of some prior knowledge about the underlying system into a parametric algorithm.

A fascinating topic that is not discussed in detail here is that of biological noise. A good review on the topic is provided by Rao et al. [67]. Briefly, a number of approaches have been developed to investigate and analyze noise in signaling and regulatory systems. Some of the recent studies include the analysis of transcriptional and translational processes to determine noise in protein populations [68], which is shown to be primarily introduced at the translational level [69]. A frequency-domain technique is described by Simpson et al. [70] and is used to analyze intrinsic noise within negatively autoregulated gene networks. These networks are shown to provide a noise filtering system that reduces the total noise variance in the protein concentration signal and shapes the noise towards high frequency. The system's properties are confirmed by experimental measurements and further analyzed to gain mechanistic insight into gene regulation [71]. In another study, Rosenfeld et al. [72] investigate the system linking transcription factor concentrations to the rate of protein production from downstream genes and characterize the transfer function for the expression of a single gene, while Pedraza and van Oudenaarden [73] study the input/output relationship of genetic circuits in order to understand noise propagation in gene networks. The reader is referred to the original papers as well as the perspective by Isaacs et al. [74] for more detail. An extensive dataset was also recently generated by Newman et al. [75]. It provides protein level information at the

single-cell level and allows the identification of protein-specific differences in noise that correlate with a protein's mode of transcription and its function.

3.3.1 Gene Regulation Systems

We start by examining the systems underlying gene regulatory networks. Specifically, the level of expression of each gene in the genome is controlled by regulatory proteins (transcription factors) that determine how effectively a given gene is transcribed to RNA, which then gets translated into a protein. Transcription factors, activated by upstream signals, interact directly or indirectly with the DNA sequence of the particular gene they regulate. They also bind each other to form multiprotein complexes, allowing fine tuning of DNA transcription. Some genes encode transcription factors allowing both feedforward and feedback loops, which generate interesting behavior and dynamics. The systematic study of gene regulatory networks was enabled by the development of gene expression microarray technology, which provides snapshots of a cell's gene expression state at different time points and under different conditions. In the next sections we provide an overview of gene expression microarray technology in order to give the reader a feel for the data collection process. We then present an example of gene regulation systems identification and analysis using signal processing techniques.

3.3.1.1 Gene Expression Microarray Technology

DNA microarray technology was first introduced in the early 1990s [76, 77] and is based on the complementarity of DNA (RNA), that is, the base-pairing of adenosine (A) with thiamine (T) (or uracil (U) in the case of RNA) and guanine (G) with cytosine (C). The microarray allows the matching of unknown DNA (or RNA) to known DNA (RNA) samples using base-pairing. In a microarray, DNA complementary to genes of interest is generated and laid out in microscopic quantities on solid surfaces at defined positions. The DNA (RNA) from samples is then eluted over the surface and complementary DNA binds. The presence of bound DNA (RNA) is then detected using fluorescence following laser excitation. The major applications of DNA microarray technology have been the identification of sequence (gene, gene mutation) and the determination of expression level (i.e., abundance) of genes. There are two main variants to DNA microarray technology, one developed by Affymetrix, Inc. and the other developed by Stanford University. The two main differences between these technologies are how DNA sequences are laid down and their length. In the Affymetrix approach, photolabile agents and photolithography techniques similar to the ones used for traditional semiconductors are used to lay out an array of oligonucleotides, while robots are used to spot glass slides at precise points with complete genes or expressed sequence tags (EST) sequences in the Stanford approach. Microarrays allow the qualitative measurement of relative expression levels of genes. A differential expression readout is obtained by using simultaneous, two-color fluorescence hybridization. In this method, fluorescent probes are prepared from two RNA sources to be compared, one labeled green and the other labeled red. Probes are mixed and washed over the microarray. Each

probe is then excited using a laser and its fluorescence at each element is detected using a scanning confocal microscope. The ratio between the red and green signals is subsequently calculated for several array elements containing total genomic DNA, which allows the detector to be calibrated such that these elements have a measured intensity ratio of 1.0. Relative intensity of the RNA probes gives a reliable measurement of the relative abundance of specific RNA in each sample, therefore detecting expression level.

The advancement of DNA microarray technology as well as genome sequencing makes it now possible to detect gene expression levels on a genomic scale under many different experimental conditions as well as at different time points. The data is usually aggregated into a $N \times M$ matrix, where N is the number of genes investigated and M is the number of experiments (or equivalently arrays) performed. The image of such an aggregate array is shown in Figure 3.16. The corresponding data matrix contains the logarithm of the relative expression level of each gene under the corresponding experimental condition (or time point). The data is then usually subjected to a series of normalizations so that the expression values of each of the genes have a mean of zero and a variance of unity across experiments.

DNA is transcribed into RNA, which is then translated into proteins, the molecular machines that carry out cell signaling and transformations. As a result, gene expression data that measure RNA abundance can be used to infer networks of co-expressed and coregulated genes, which can then be mapped into the corresponding protein networks [78]. Alternatively, expression data obtained from mutant experiments allow the construction of genome-wide disruption networks [79].

3.3.1.2 Gene Expression Signal Definition

A natural definition of signals here is gene expression. In particular, the signals we are interested in, denoted by $x_i[m]$, give the relative gene expression level of a gene

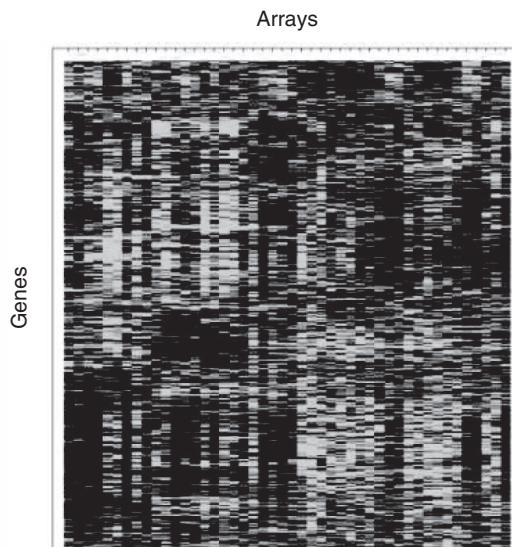


Figure 3.16 Example of data clustered from a DNA microarray.

of interest, i , under experimental condition m . In temporal studies, the experimental condition represents different time points and therefore m refers to time samples. In other studies, the experimental conditions can be different temperatures or different drug concentrations, in which case m is defined accordingly. Since there are M different experimental conditions and N total genes, $x_{i[m]}$ is defined for $0 \leq m \leq M - 1$ and $0 \leq i \leq N - 1$. The total data captured by the microarray experiment can therefore be synthesized into an $N \times M$ matrix as follows:

$$X \equiv \begin{bmatrix} x_{0[m]} \\ x_{1[m]} \\ \vdots \\ x_{N-1[m]} \end{bmatrix}$$

The rows of X correspond to the *gene's transcriptional responses* while the columns of X correspond to the *array's expression profiles*.

A wide range of mathematical techniques have been recently applied to gene expression data, essentially involving manipulations of the matrix X or its columns or rows. For example, self-organizing maps have been applied to hematopoietic differentiation expression data in order to cluster the data [80], while singular value decomposition has been used to transform expression data from a *genes $\times$ arrays* space into a reduced “*eigengenes*” $\times$ “*eigenarrays*” space where analysis is further carried out [81, 82]. Networks have also been inferred from gene expression data and the spectra of the resulting graphs have been analyzed [78, 79, 83], while Fourier analysis has been applied to time measurements of gene expression [84, 85]. We next discuss the singular value decomposition technique described by Alter et al. [81, 82] and provide a brief highlight at the end of the section of the Fourier analysis technique described in [85]. The reader is referred to the original papers for detailed information about the methodologies and results.

3.3.1.3 Gene Regulation System Identification Using Singular Value Decomposition

Singular value decomposition (SVD) and principal component analysis (PCA) are powerful linear algebraic techniques that have been used in signal processing problems including image processing, speech processing, compression, and systems and signals modeling and detection. They represent simple nonparametric methods that allow the extraction of relevant information and structure from noisy measurements, which can sometimes lead to dimension reduction and the identification of hidden and simplified structure underlying the data. PCA can be obtained through specific interpretations of the SVD. We therefore focus here on the SVD and occasionally provide interpretations to connect it to PCA.

Singular value decomposition essentially decomposes any $N \times M$ matrix X into the product of two orthonormal matrices and a pseudodiagonal matrix. The orthonormal matrices are an $N \times N$ matrix U and an $M \times M$ matrix V^T ($U^T U = I_N$ and $V^T V = I_M$, where I_N and I_M refer to the $N \times N$ and $M \times M$ identity matrices respectively) and the pseudodiagonal matrix is an $N \times M$ matrix S , where all entries are

zero except for the first $L \equiv \min(M, N)$ diagonal entries. These diagonal entries are referred to as singular values and denoted by $\sigma_1, \sigma_2, \dots, \sigma_L$. They are usually ranked in descending order by convention (i.e., $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_L$). The matrix X can therefore be written as

$$X = USV^T \quad 3.20$$

The first L columns of V , $v_1, v_2, \dots, v_L$ (which correspond to the first L rows of V^T) are the orthonormal eigenvectors of the symmetric matrix $X^T X$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_L$. The singular values can be derived from the eigenvalues of $X^T X$, where $\sigma_i = \sqrt{\lambda_i}$. The first L columns of U , $u_1, u_2, \dots, u_L$, are obtained from the columns of V as follows: $u_i = \frac{1}{\sigma_i} X v_i$. The remaining $M-L$ and $N-L$

column vectors of V and U respectively are obtained by picking any set of additional orthonormal vectors. Since the associated singular values are zero, the specific choice of these vectors does not matter and they can be calculated using any extension method such as the Gram-Schmidt orthogonalization.

Equation (3.20) has a number of important interpretations. It can be rewritten as

$$U^T X = S V^T$$

where the change of basis from X to $S V^T$ performed by U^T becomes explicit. Stated differently, the columns of U represent a basis for the column space of X . The equation can also be rewritten as

$$V^T X^T = S U^T$$

which explicitly shows the change of basis from X^T to $S U^T$ performed by V^T (i.e., the columns of V represent a basis for the column space of X^T or equivalently the row space of X). Orthonormal bases that span the column space of a normalized version of X correspond to the principal components of X in PCA, that is, the columns of U correspond to the principal components of X after proper normalization. For a more detailed discussion of singular value decomposition and principal component analysis, the reader is referred to [86, 87].

In the case of gene expression data, the elements of the i^{th} row of the data matrix X , $x_i[m]$, correspond to the transcriptional response of gene i , that is, the normalized gene expression level of gene i for the experimental conditions (arrays) m , while the elements of the j^{th} column, $y_j[n]$, correspond to the expression profile of the j^{th} array, that is, the normalized expression levels of genes n for experimental condition m .

Applying the singular value decomposition to the gene expression data matrix allows the transformation of the data from the $N\text{-genes} \times M\text{-arrays}$ space to the reduced $L\text{-"eigengenes"} \times L\text{-"eigenarrays"}$ space obtained from the first L row vectors of V^T and the first L column vectors of U respectively. This technique essentially transforms genes into eigengenes such that the l^{th} eigengene is expressed only in the corresponding l^{th} eigenarray with a corresponding "eigenexpression" level σ_l (the l^{th} entry of S). The expression of each eigengene (eigenarray) is therefore decoupled from that of all other eigengenes (eigenarrays). The relative strength

of eigenexpression is given by $\sigma_l^2 / \sum_{k=1}^L \sigma_k^2$ and indicates the relative significance of the l^{th} eigengene and eigenarray in terms of the fraction of the overall expression that they capture. This transformation allows the representation of the transcriptional response of the n^{th} gene as a linear combination of L eigengenes and the representation of the expression profile of the m^{th} array as a linear combination of L eigenarrays. Equivalently, the transcriptional response, $x_i[m]$, could therefore be studied in the reduced space of eigengenes instead of the higher-dimensional genes space. The axes in this space are given by the first L columns of V , $v_1, v_2, \dots, v_L$, and the coordinates of the genes in this space are given by the rows of the matrix US . Alternatively, one can study the data in the array space, that is, the expression profiles, $y_i[n]$, can be studied in the reduced space spanned by the L columns of U , $u_1, u_2, \dots, u_L$, which represent an orthonormal basis for this space. The coordinates of the individual arrays in this space are given by the columns of the matrix SV^T . Subject to proper normalization, this result corresponds to the PCA result where $u_1, u_2, \dots, u_L$ represent the principal components and the elements of SV^T are the principal components' scores. In both representations the basis vectors are weighted by the singular values. Dimensionality reduction can therefore be achieved by discarding basis vectors associated with small singular values. This application is sometimes referred to as the dimensionality reduction application of singular value decomposition and principles component analysis.

The SVD of X therefore produces two sets of orthonormal bases. The first basis is defined by the right singular vectors, $v_1, v_2, \dots, v_L$, which span the row space of X , that is, the space spanned by the gene transcriptional responses. The second basis is defined by the left singular vectors, $u_1, u_2, \dots, u_L$, which span the column space of X , that is, the space spanned by the array expression profiles. Examining the data in these spaces and analyzing the associated basis vectors provides great insight into the system underlying gene regulation.

Gene expression profiles obtained from the budding yeast *Saccharomyces cerevisiae* were generated by Spellman et al. [84] and examined by Alter et al. [81]. Specifically, genome-wide mRNA levels of yeast genes in a yeast culture synchronized by alpha factor monitored over approximately two cell cycle periods relative to a reference mRNA from an asynchronous yeast culture were examined. The data was sampled at 7 min intervals for 119 min. Two additional sets of experiments were also included where mRNA levels of yeast strain cultures with overactivated CLB2 (which encodes a G2/M cyclin) and CLN3 (which encodes a G1/S cyclin) genes were measured at two different time points relative to their levels at the start of the overactivation at $t = 0$. The data was then normalized, prefiltered, and assembled into a $4,579 \times 22$ data matrix obtained from collecting the expression level of 4,579 genes using 22 arrays (corresponding to 7-min sampling of two cell cycle periods ($t \sim 119$ min) as well as 4 arrays corresponding to the CLB2 and CLN3 treatments). The SVD of the data matrix was computed to derive the eigengenes and eigenarrays. The resulting first two eigengenes corresponding to the two largest singular values had similar significance and together captured 40% of the overall normalized expression. These dominant eigengenes were found to closely approximate sine and cosine functions, possibly representing cell cycle expression oscillations. Specifically, the first two eigengenes fit normalized sine and cosine

functions of two 66-min periods during the cell cycle (from $t = 7$ to 119 min). For the CLB2 and CLN3 experiments, the second eigengene described steady-state expression, while the first eigengene described underexpression in the CLB2-overactive arrays and overexpression in the CLN3-overactive arrays. Figure 3.17 shows these eigengenes.

Projecting all 4,579 genes onto the subspace spanned by the first two eigengenes provides a method to identify genes that are cell cycle regulated. Specifically, genes that have almost all of their normalized expression in this subspace (i.e., their distance from the origin is close to 1) are cell cycle regulated, whereas genes that have almost no expression in this subspace (i.e., those close to the origin) are not expected to be regulated by the cell cycle. In fact, it is found that most of the genes that have a significant amount of their normalized expression in this subspace correspond to the genes that have been identified by Spellman et al. as cell cycle regulated, therefore validating the methodology. The correlation of eigengenes with genes known to peak at certain stages of the cell cycle was then examined, allowing a classification of the eigengenes. Specifically, positive correlation with the first eigengene corresponds to cell cycle expression oscillations that start at the transition from G1 to S1 and are dependent on CLN3, while negative correlation with the first eigengene is associated with oscillations that start at the transition from G2/M to M/G1 and are dependent on CLB2. Positive correlation with the second eigengene is associated with oscillations that start at the transition from M/G1 to G1, while negative correlation with the second eigengene is associated with oscillations that start at the transition from S to S/G2 as shown in Figure 3.18. The 22 arrays can also be projected onto the subspace spanned by the first two dominant eigenarrays corresponding to the two dominant eigengenes as shown in the figure. It is observed that sorting the arrays according to their phases (i.e., according to their transition from the expression pattern of one eigenarray to the other) gives an array order that corresponds to the cell cycle time points measured by the arrays and therefore de-

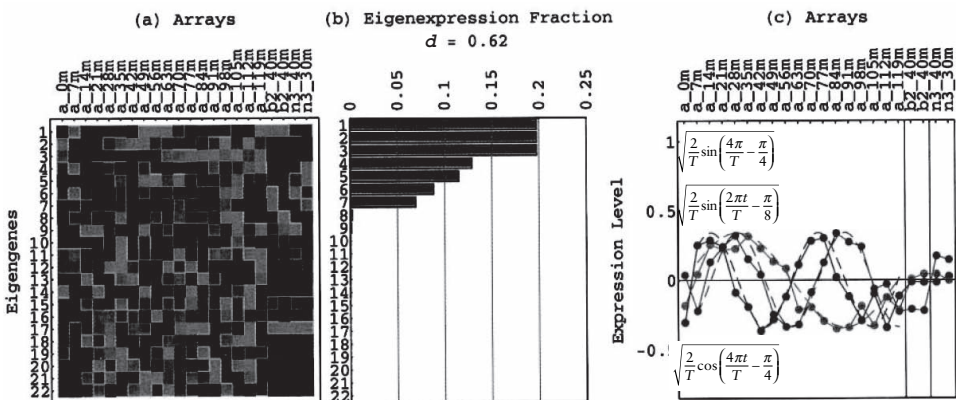


Figure 3.17 (a–c) Eigengenes resulting from applying the SVD to normalized expression data from 4,579 yeast genes. Genes were collected at 18 different points during two cell cycle periods of yeast culture synchronized by alpha factor as well as 4 arrays obtained from experiments of two yeast strain cultures with overactivated CLB2 and two yeast strain cultures with overactivated CLN3. (From [81].) © 2004 National Academy of Sciences, U.S.A.

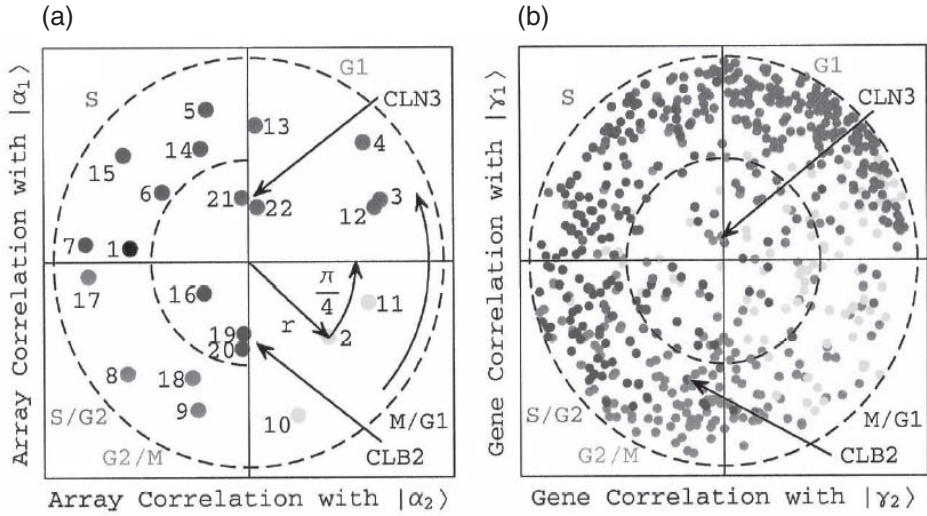


Figure 3.18 Normalized alpha factor, CLB2, and CLN2 expression data projected on the subspace spanned by the first two eigenarrays (a) and the first two eigengenes (b). (Reproduced from [81].) © 2000 National Academy of Sciences, U.S.A.

scribes the progression of the cell cycle expression across the different stages. These dominant eigenarrays therefore capture all cell cycle cellular states.

A number of other studies have explored the use of SVD to analyze biological systems using gene expression data. For example, in a subsequent study, Alter et al. [82] use a generalized form of singular value decomposition to compare data from humans and yeast. They compare the different eigengenes between these species, which allows the identification of processes specific to the different species and others common between the two species. In other applications such as diagnostic applications, where the problem consists of classifying tissue samples from individuals with and without a disease, the focus is mainly on the space spanned by the eigenarrays. In addition to singular value decomposition, a number of signal processing techniques have been proposed to analyze and identify gene regulatory systems. In particular, Butte et al. [85] identify candidate pairs of genes related by phase-shifts by applying Fourier analysis (specifically power spectral density analysis) to time-series gene expression data. They consider pairs of genes as input/output pairs connected via a biological system. The LTI property of the biological system is investigated by computing an estimate of the coherence function, which is defined as the ratio of the magnitude squared of the cross-spectral density between the input and the output and the product of the power-spectral densities of the input and output as follows:

$$C_{xy}(\omega) \equiv \frac{|P_{xy}(\omega)|^2}{P_{xx}(\omega)P_{yy}(\omega)}$$

where x and y are the input and output signals respectively and $P(\cdot)$ denotes the appropriate power spectrum. A coherence value close to one is indicative of an LTI system, while a low coherence indicates that the two signals are not linearly related.

The transfer functions of the systems with high coherence are then computed by taking the ratio of the cross-spectral density over the power spectral density of the input. The magnitude and phase are investigated. Using this methodology, expression profiles from 2,467 genes obtained from yeast exposed to α -pheromone and sampled at 7-min intervals 17 times were analyzed. The analysis allowed the identification of 18 gene-gene associations with high coherence and gain at frequencies with a phase-shift of 5 min, two of which have already been well known to be associated. This method therefore provides a way to identify interactions among genes whose expression profiles are not synchronized due to, for example, the time needed for activity to take place. However, care should be taken when performing this analysis since the limited number of samples available and the nonuniform sampling across time points may limit the information content as well as spectral resolution attainable by the algorithm and therefore significantly reduce the signal-to-noise ratio.

3.3.2 Protein Signaling Systems

As shown in the previous section, identifying and analyzing the systems underlying gene regulation provide insight into the wiring of cellular circuits and are therefore an indication of the spectrum of possible behaviors that a given cell may exhibit. A number of events, however, are not visible at the level of gene regulation. In particular, intracellular signaling networks composed primarily of proteins and enzymes process signals on shorter time frames than those needed to complete transcription and translation of a given gene. In addition, these signaling events do not necessarily induce changes in gene expression and therefore may be totally invisible at the gene regulation level. Protein signaling typically occurs through a series of protein modifications such as phosphorylation (the addition of phosphate groups) or cleavage as well as translocation of proteins and enzymes across cellular compartments (such as from the cytoplasm to the nucleus). In this section, we discuss two models aimed at identifying and analyzing the systems underlying intracellular signaling. We first describe a system identification technique, similar to the one described for gene expression, which uses singular value decomposition [88, 89]. We then focus on system analysis and describe a technique for analyzing protein signaling networks using Fourier transforms [90].

In addition to the techniques described here, a number of other signal processing approaches have been developed and applied to identify and analyze protein signaling systems. In a series of studies, Korokova et al. [91, 92] used spectral analysis of biological noise in individual cells to investigate temporal behavior variability in the bacterial chemotaxis network. In another study, the signal processing performed by the TGF- β signaling pathway was investigated [93]. Specifically, depending on the cell type, members of the TGF- β superfamily can lead to different outputs, including inhibition of growth, differentiation, apoptosis, cell migration, and adhesion. The signal processing system that governs the first layer of processing around the TGF- β receptor is analyzed, and it is demonstrated that the system is capable of generating rich input/output characteristics, therefore explaining the wide output behavior experimentally observed for TGF- β . The reader is referred to the original papers for more detail.

3.3.2.1 Protein Signal Definition

Unlike gene regulation systems, which have been interrogated primarily by way of expression profiling experiments, different experimental techniques have been used and are being developed to study protein signaling systems. This apparent heterogeneity of data collection is due to the disparity of different protein signals as well as to the inherent difficulty in measuring them. Defining protein signals is therefore not as straightforward as defining gene signals, which in most cases are the result of one class of experiments, namely expression profiling.² Protein signals are often measured using different experimental techniques. For example, western blots and protein arrays measure protein abundance or levels, while enzyme assays investigate enzymatic activity by measuring the production of product or consumption of substrate. Other experiments such as yeast-two-hybrid systems and some based on mass spectrometry measure interactions among different proteins. The fusion of heterogeneous protein measurements is therefore an important challenge when studying protein signaling systems. In a recent study, Gaudet et al. [94] provide a good discussion of the challenges associated with heterogeneous data. They also describe a methodology for fusing ~10,000 measurements obtained using different techniques.

Defining the signal of interest is therefore dependent not only on the biological question under investigation but also on the experimental data that is accessible through the different measurement techniques. In this section, we present two examples of signal processing techniques applied to biological systems that use two different representations of protein signals: a discrete-time representation and a continuous-time representation. In the first example, protein signals are defined as discrete-time variables very much like the gene expression signals in the previous section. However, instead of focusing on a uniform readout such as expression level for the gene expression signals, different properties are used to define the different protein signals depending on the measurement readout. Specifically, the protein signal in this case, $x_i[n]$, refers to a certain property such as the total amount or the activity level of a given protein i of interest measured under a certain experimental condition n . The experimental condition in the example we present typically refers to a certain input level. Temporal measurements are either considered as different experimental conditions (as was the case in the gene expression data) or can be used to define new signals such as the activity of a given protein at specified time points or the peak activity of a given protein (referred to as metrics in the example), in which case the experimental condition would only refer to input level. In the second example we discuss, a continuous-time representation of protein signals is used. In this case, protein signals refer to concentrations of particular proteins as a function of time. We use the notation, $x_i(t)$, to refer to the concentration of protein i at time t in this case. In addition to proteins, a number of molecules such as ligands and hormones play an important role in protein signaling systems. These molecules are treated in a similar way as the proteins discussed above.

2. There are a number of other experimental techniques that interrogate gene regulation systems; however, these are usually lower throughput and therefore less widely used for modeling using signal processing techniques.

3.3.2.2 Apoptosis System Identification

In the previous section, we presented a method using singular value decomposition that allowed the transformation of gene expression data from a *gene* $\times$ *arrays* space into a reduced “*eigengenes*” $\times$ “*eigenarrays*” space that provided insight into the underlying gene regulation system. Multivariate techniques have also recently proved useful for analyzing and identifying protein signaling systems. However, while data amenable to multivariate analysis is readily available for gene regulation systems through high-throughput gene expression arrays, obtaining multivariate data at the protein signaling level has proved more challenging due to the inherent difficulties of measuring different protein states and the heterogeneity of the data collected. Recently, an extensive dataset was generated by Janes et al. [88, 89] in which 7,980 distinct molecular signals underlying the programmed cell death signaling network were experimentally measured. The data was analyzed using multivariate techniques to understand the signal processing performed by this network.

Programmed cell death, also referred to as apoptosis, is the process by which certain signals lead cells to self-destruct. The signaling system underlying apoptosis allows the cell to process input signals capturing information coming from the environment of the cell to lead to one of two possible outputs: cell survival or cell death. Input signals are typically encoded in soluble proteins such as the pro-death cytokine tumor necrosis factor alpha (TNF) or pro-survival growth factors such as epidermal growth factor (EGF) or insulin. The system output is typically a phenotypic readout (death or survival); however, it can also be determined by measuring “early” signals that perfectly correlate with the death/survival output. Examples of such early signals include phosphatidylserine exposure, membrane permeability, nuclear fragmentation, and caspase substrate cleavage. Figure 3.19 illustrates the proteins involved in the TNF-EGF-insulin apoptosis signaling system as well as the different experimental techniques used to probe them. Figure 3.20 shows the system

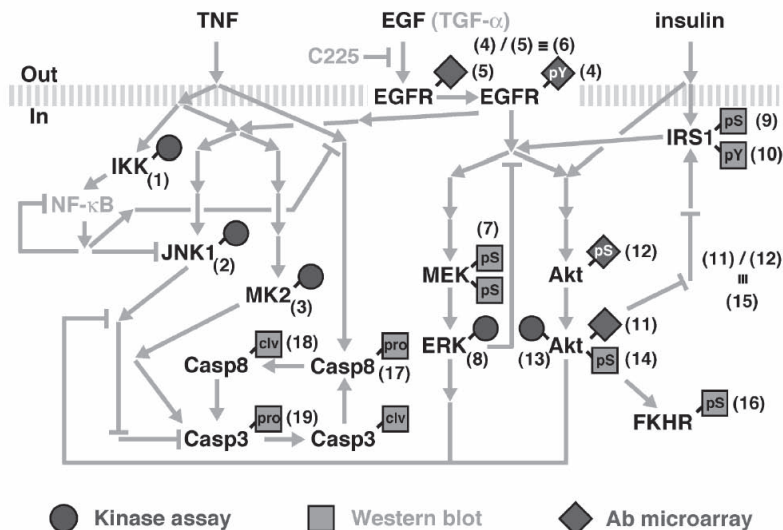


Figure 3.19 Schematic of the shared TNF-EGF-insulin signaling network. The 19 molecular signals (numbered) from protein nodes were measured by high-throughput kinase assays, quantitative Western blotting, or antibody microarrays. (From [89].) Reprinted with permission from AAAS.

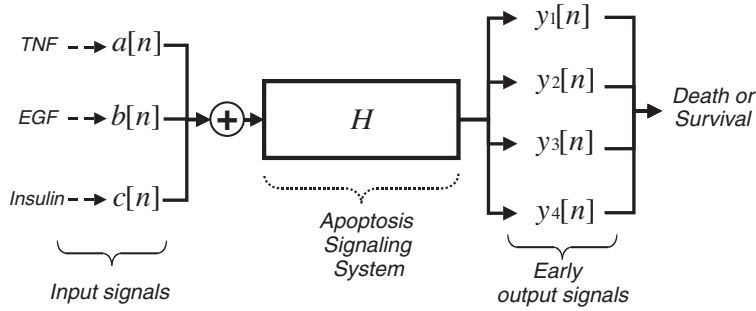


Figure 3.20 Illustration of the system under investigation (in Janes et al. [89]).

under study where the input signals are denoted by $a[n]$, $b[n]$, $c[n]$ and the early output signals are denoted by $y_1[n]$, $y_2[n]$, $y_3[n]$, $y_4[n]$.

In order to identify and analyze the apoptosis system, H , in Figure 3.20, one has to first identify the components (or building blocks) that are important in defining the system function and then develop a model for the system function. A series of experimental measurements was therefore carried out on potential system components (referred to as molecular signals) and on the output signals obtained in response to a combination of input signals. The measurements of the system components in response to a set of inputs were then analyzed using SVD and PCA in order to identify the combination of components that is most important in processing the input, that is, the building blocks of the system function. A model for the system function was subsequently defined using partial least squares regression.

In order to identify the system components that are most important in processing the signal, an approach similar to the one used to identify the gene regulation system described in the previous section was used. Specifically, in a first set of experiments [88], 29 protein signals were identified, including various kinases activity such as the activity of the kinases ERK and JNK as well as cleavage states of different caspases such as caspase 8 and caspase 3. The 26 experimental measurements were collected on these signals, resulting from 13 time points measured for two different sets of inputs: a TNF input and a combination of TNF and insulin input. The resulting data matrix, X , has therefore 29 rows representing the protein signals and 26 columns representing the experimental measurements.³ Applying the SVD to the data matrix X allows the transformation of the protein signaling data from a *protein* $\times$ *experiments* space into a reduced “*eigenproteins*” $\times$ “*eigenexperiments*” space analogous to the “*eigengenes*” $\times$ “*eigenarrays*” space discussed in the previous section.

The principal components of X correspond to the columns of U produced by the SVD of the normalized data matrix.⁴ They allow the identification of independent vectors composed of linear combinations of protein signals that best capture

3. In order to be consistent with the previous section, we use a data matrix that is the transpose of the matrix used in Janes et al., where the rows of the data matrix represent the experiments while the columns represent the signals.

4. See the section on gene expression for a description of the different matrices involved in SVD.

the variance in the experiments space (i.e., the column space of X). It is observed that the first two principal components capture more than 60% of the total variance, allowing one to analyze the experiments in the reduced two-dimensional space spanned by these first two principal components. Figure 3.21, reproduced from [88], shows a projection of the data onto this two-dimensional space. A temporal segregation of the data is achieved, where later time points (illustrated by the large diamonds and squares in the figure) correspond to points with relatively small coordinates along the first principal component and large coordinates along the second principal component. In addition, highly significant segregation of the data resulting from the two different inputs (TNF only and TNF+insulin) was achieved. This two-dimensional space therefore captures most of the processed information, that is, the system components defined by the two principal components capture most of the building blocks used to process the input. Close investigation of the protein signals represented in the first two principal components (i.e., the change of variable given by the matrix U) identified a “pro-survival” component composed primarily of signals that have been previously associated with pro-survival pathways such as the phosphorylated kinases p-Akt and p-IkB and a “pro-death” component composed primarily of signals associated with pro-death pathways such as cleaved caspases. Furthermore, while more than one signal was needed to define a given principal component, not all protein signals were needed, confirming the existence of redundancies in the measured dataset. These results suggest that a good approximation of the underlying processing system can be based on two dimensions composed of a subset of pro-death and pro-survival components (protein signals).

The results were further confirmed and refined by analyzing a more extensive experimental dataset based on 660 metrics derived from time course measurements of 19 molecular signals [89]. The values of these 660 metrics were collected for 9 different experimental conditions corresponding to different combinations of the three input signals shown in Figure 3.20 and resulting in a 660×9 data matrix. As for the earlier results, the first two principal components of this data matrix captured most of the variance in the experimental measurements and allowed segregation of pro-death and pro-survival signals, confirming the relevant system components.

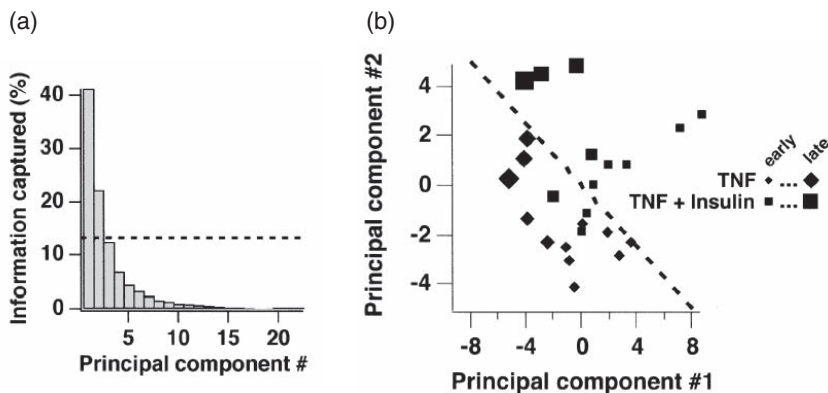


Figure 3.21 (a, b) Principal component analysis performed on data from the apoptosis signaling pathway identifying two dimensions along which most of the data could be analyzed. (From: [88].) © 2004 Mary Ann Liebert, Inc. Reprinted with permission.

Understanding the components important for signal processing provides insight into the architecture of the apoptosis system; however, it does not provide a description of the system, that is, how the different components interlink to generate the system response. Specifically, computing and analyzing the principal components of the data matrix X is not sufficient to provide an input/output description of the system, since information captured by the output signals is not incorporated into the analysis. A description of the input/output relationship can, however, be obtained through a simple extension of the previous analysis using partial least-squares regression (PLSR). Specifically, an output matrix is defined capturing the output signal under each experimental condition. In the first study [88], a binary output is used where death or survival is determined from the input signal (TNF is a death signal whereas insulin is a survival signal). In the second study [89], the output signals are measurements of the early outputs shown in Figure 3.20.

PLSR is similar to principal component regression in that a set of coefficients is computed to best (in a mean squared sense) predict the output based on a set of orthonormal vectors that describe the data. However, unlike principal component regression, which uses as orthonormal vectors the principal components of the data matrix X , PLSR computes a set of orthonormal vectors that capture the most covariance between the output matrix and the data matrix, therefore analyzing the data along the dimensions that contain the most input/output information. Using PLSR, a second-order linear system that describes the system function H was identified. This system was validated by successfully predicting outputs in response to a new set of inputs not used in the system identification. It was also used to identify new molecular mechanisms connecting autocrine circuits to apoptosis, providing additional insight into the signal processing performed by this system. The reader is referred to the original papers for more details.

3.3.2.3 Biochemical Reactions System Analysis

Analyzing protein signaling systems using multivariate techniques such as the SVD, PCA, and PLSR discussed above provides useful ways to identify relevant components and determine a model for the system function without prior knowledge of the underlying system details. These so-called nonparametric methods are particularly useful when the underlying system is not known. Often, however, we do have some knowledge of the underlying biology, that is, of the individual building blocks and how they are interconnected. Capturing this knowledge in our system identification and analysis approach is therefore useful and desirable. Here we discuss a set of techniques that capture prior biological knowledge regarding the interconnections of biochemical reactions governing protein signaling systems.

Protein signaling networks are typically composed of a collection of biochemical reactions that can be modeled using some mathematical formalism. Consider a volume V of a cell or compartment containing a mixture of N species (proteins, chemicals, or other molecules) that can interact through M specified biochemical reactions. Given the number of molecules of each species present at some initial time, the modeling problem consists of finding the molecular population levels at any later time in the volume V . Several approaches have been formulated to solve this problem. Different approaches make different assumptions in order to make the

problem more tractable and simplify the analysis. Early approaches relied on deterministic formalisms while, more recently, stochastic approaches have been proposed and successfully implemented using a direct algorithm, a Petri net formalism [95], as well as Markov-modulated Markov chains [1, 96].

A common approach to modeling biochemical reaction networks consists of translating each reaction involved in the network into a set of ordinary differential equations. This deterministic approach is based on two basic assumptions: the number of molecules of each species can be represented by a continuous single-valued function and each of the M biochemical reactions can be regarded as a continuous-rate process. Also, the volume V of the cell is generally assumed to be fixed and the species mixture is assumed to be uniform. Using these assumptions one can easily construct a set of coupled, first-order, ordinary differential equations of the form

$$\frac{dX_i}{dt} = f_i(X_1, \dots, X_N)$$

where $i = 1, \dots, N$ and N is the total number of species. The specific forms of the functions f_i are determined by the structures and rate constants of the M biochemical reactions. These equations express the time-rate-of-change of the molecular concentration of one chemical species, X_i , as a function of the molecular concentrations of all the species, $X_1, \dots, X_N$. They are termed the reaction-rate equations and their solution gives the time evolution of the network. In large volumes, the rate of the reaction is proportional to the concentration of the reactants. This is the Law of Mass Action, which has been well established in large volumes based on experiments that date back to the mid-1800s. In the early 1900s, Michaelis and Menten published a model for enzyme kinetics where an enzymatic reaction is written as



where S and P are the substrate and product respectively and E and ES are the free enzyme and the enzyme-substrate complex respectively [97]. The Law of Mass Action applied to this basic enzymatic reaction leads to a set of coupled differential equations that can be approximated using perturbation theory or solved numerically.

Signal processing can be used to analyze biochemical reaction networks. Specifically, most signaling networks can be decomposed into a collection of elementary first-order unimolecular and second-order bimolecular reaction steps. In a first-order reaction, the reaction rate is proportional to the concentration of one of the reactants, while in a second-order reaction the reaction rate is proportional to the concentration of the square of a single reactant or the product of the concentrations of two reactants. As a simple example, consider the following first-order reversible chemical reaction, where species X is converted to species Y at rate k_1 and Y is converted back to species X at rate k_2 :



This reaction is described by the following first-order differential equation:

$$\frac{dy}{dt} = k_1x(t) - k_2y(t) \quad 3.22$$

where $x(t)$ and $y(t)$ denote the concentrations of species X and Y respectively as a function of time. This equation may be obtained from an enzymatic reaction following Michaelis-Menten kinetics, where the substrate is present at concentrations significantly less than the Michaelis constant. Since linear constant-coefficient differential equations describe LTI systems, the equation above can be modeled by an LTI system linking species X to species Y within the framework presented in Figure 3.1. Specifically, the frequency response of the corresponding first-order system, $H(j\omega)$, linking X to Y, is

$$H(j\omega) = \frac{k_1}{j\omega + k_2} \quad 3.23$$

This system represents a low-pass filter, that is, the amplitude of the frequency response decreases with increasing frequency. Figure 3.22 shows the plot of the magnitude and phase of $H(j\omega)$ for $k_1 = 0.5$ and $k_2 = 0.25$.

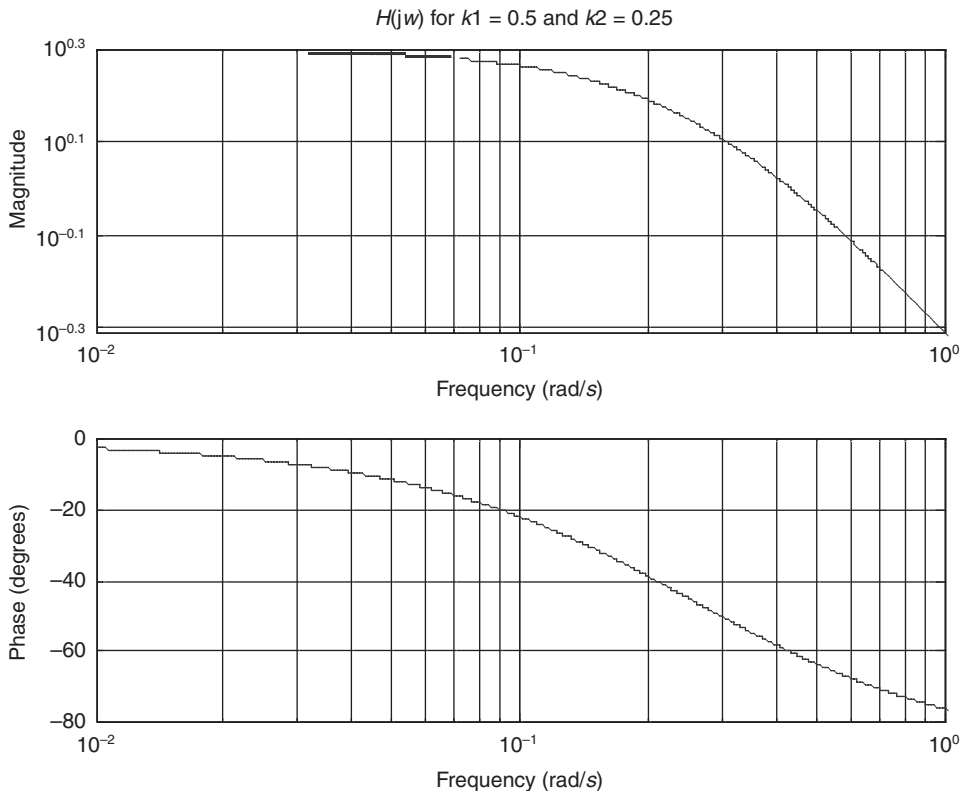


Figure 3.22 Magnitude and phase of the system representing a first-order reversible chemical reaction with forward rate $k_1 = 0.5$ and backward rate $k_2 = 0.25$.

A generalization of this approach is considered by Samoilov et al. [90], where a number of chemical reaction networks are analyzed for their response to periodic single- and multifrequency signals. Specifically, consider the simple extension to the reaction presented above, where each species is allowed to enter and exit the compartment at some given rate as shown in Figure 3.23. Samoilov et al. show that given selected choices of oscillatory inputs $i_x(t)$ and $i_y(t)$, the system can be made to exhibit interesting behavior. In particular, a bandpass filter can be implemented if both species are independently driven. More generally, if the system is extended to include a network of n species coupled by first-order chemical reactions with basic building blocks as the one shown in Figure 3.23, then a bandpass filter can be implemented if at least two species are independently driven. Furthermore, the linear system can only be a low-pass filter if it is driven by a single oscillatory input.

Understanding the properties of these systems provides a means to guide system identification. For example, if experimental data suggest bandpass behavior in response to a single oscillatory input, then the results show that the underlying system has to contain nonlinear reactions. In fact, while no general solution exists in the case of nonlinear chemical reaction networks (such as bimolecular reactions steps), it is shown that for specific examples a band-pass filter is always attainable. Cells therefore seem to be capable of differentially processing signals by using different, relatively simple, biochemical networks. Frequency-modulated signals can therefore be processed by a bank of frequency-selective filters implemented through biochemical reactions as shown in Figure 3.24. Hormone-induced calcium release from internal stores into the cytosol provides an example of a frequency-modulated signal leading to the activation or deactivation of a number of different pathways. For example, it has been shown that the systems underlying neuronal differentiation differentially process frequency modulated calcium signals [98]. In particular, different waveforms activate different pathways such as neurotransmitter expression, channel maturation, and neurite extension. Relatively high-frequency calcium signals were shown to be important to achieve sustained activation of mitochondrial metabolism through activation of calcium-sensitive mitochondrial dehydrogenases [99]. Active modulation of intracellular calcium signals through different biological mechanisms using various types of signals has also been demonstrated in a number of different studies. For example, Wagner et al. [100, 101] showed that both extracellular calcium signals as well as calcium signals released from internal stores are needed for proper frequency modulation of the cytosolic calcium signal in rat pituitary cells. Frequency and amplitude modulation of the cytosolic calcium signal induced by two different signals, a thyrotropin-releasing hormone and a calcium channel agonist, was also demonstrated, providing insight into how cells preprocess

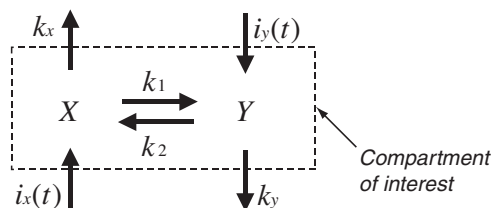


Figure 3.23 A two-species-driven linear equation.

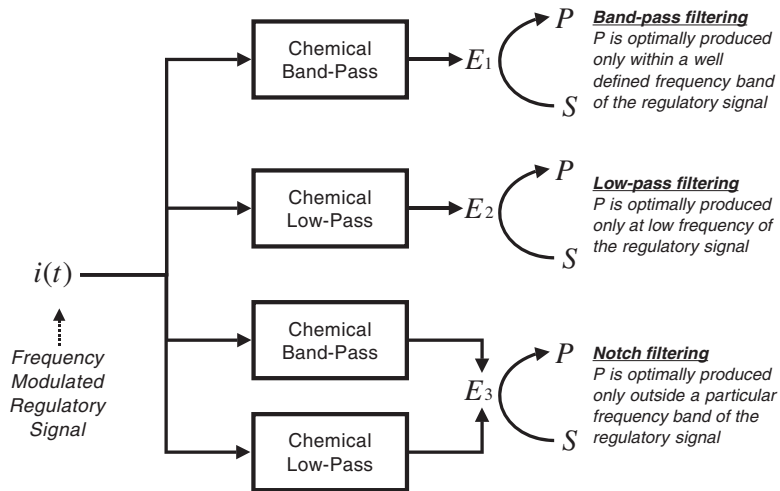


Figure 3.24 Schematic diagram for three different chemical filters of a single signal. (After [90].)

signals to control cell function. Similarly, Costantin and Charles [102] demonstrated that specific subtypes of potassium channels play a distinct role in the frequency or amplitude modulation of calcium signaling in neuroendocrine cells, suggesting a role for these channels in influencing cellular activity through shaping the calcium signal.

3.4 Conclusion

This chapter introduced biological signal processing at the cell level by providing an overview of fundamental signal processing techniques essential for the analysis of biological data, signals, and systems. A historical perspective was first provided, reviewing examples highlighting the interplay between signal processing and biology. An overview of signal processing concepts of particular interest to biology was then provided. Continuous-time and discrete-time signals were first discussed and an overview of time-domain and frequency-domain representations as well as single- and multidimensional signals was highlighted. Random processes and power spectra were then reviewed, as well as an overview of linear and nonlinear system theory with applications to biology was provided. Signal detection and estimation theory was subsequently discussed through case studies of biological problems. In particular, the problem of DNA basecalling was formulated as well as gene feature extraction and genomic signal processing. System identification and analysis was introduced in Section 3.3 at two different levels, gene expression and protein signaling, where we provided examples of some of the signal processing techniques that have been developed in these contexts.

It is important to remind the reader that we have only discussed one of the two objectives of biological signal processing as highlighted in the introduction and have focused our discussion on signal processing *within cells*. In particular, we have not addressed the efforts seeking to use biology as a metaphor to formulate novel signal processing algorithms for engineered systems. The research in this area

is very exciting. For example, with the current spread of heterogeneous networks, designing flexible signal processing algorithms that are able to adapt to the current state of the network is becoming essential. One could envision drawing ideas from the study of signaling networks to formulate adaptive distributed signal processing algorithms. In fact, cells are highly dynamic systems where, through multiple signaling cascades and shared messenger molecules, signals get processed based on enzyme availability and physical proximity. In other words, processing jobs are distributed and routed to different processors based on the state of the cell at the time the signal arrives. Furthermore, each processing element is acting autonomously without any central control. Cells seem to have therefore chosen decentralized and distributed real-time signal processing algorithms to generate cellular responses to stimuli. Identifying and deciphering these algorithms are first steps towards developing and engineering novel adaptive real-time signal processing algorithms for distributed processing environments.

The examples highlighted in this chapter strongly suggest that contributions from interdisciplinary research in the domain of signal processing and molecular biology can be made simultaneously in both directions. In other words, it is, for example, through using signal processing to model signaling pathways that one will be able to develop novel signal processing algorithms based on biological signaling. It is potentially an exciting technology opportunity because significant advances in the knowledge of how cells process signals may lead to innovations in our creation of new decision-making, computing, and signal processing systems, including devices as well as algorithms.

Acknowledgments

I wish to thank the editors Dr. Gil Alterovitz and Professor Marco Ramoni for the invitation to write this chapter. I am also grateful to Professor Alan Oppenheim and Dr. Joseph Saleh for critical comments on the manuscript and to Dr. Kevin Janes for helpful discussions on the apoptosis signaling network.

References

- [1] Said, M. R., "Signal processing in biological cells: proteins, networks, and models," Ph.D. thesis, MIT, 2005. Also available as *RLE Tech. Rep.*, No. 711, MIT, June 2006.
- [2] Bernard, C., *The Cahier Rouge of Claude Bernard*, Cambridge, MA: Schenkman Pub. Co., 1967.
- [3] Bertalanffy, L. V., *General System Theory: Foundations, Development, Applications*, rev. ed., New York: G. Braziller, 1968.
- [4] Wiener, N., *Cybernetics: Or, Control and Communication in the Animal and the Machine*, Cambridge, MA: Technology Press, 1948.
- [5] Grodins, F. S., *Control Theory and Biological Systems*, New York: Columbia Univ. Press, 1963.
- [6] Reiner, J. M., *The Organism as an Adaptive Control System*, Englewood Cliffs, NJ: Prentice-Hall, 1967.

- [7] Oppenheim, A. V., A. S. Willsky, and S. H. Nawab, *Signals & Systems*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 1997.
- [8] Oppenheim, A. V., R. W. Schafer, and J. R. Buck, *Discrete-time Signal Processing*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 1999.
- [9] Wiener, N., *Extrapolation, Interpolation, and Smoothing of Stationary Time Series, with Engineering Applications*, Cambridge, MA: Technology Press of MIT, 1949.
- [10] Vetterli, M., and J. Kovacevic, *Wavelets and Subband Coding*, Englewood Cliffs, NJ: Prentice Hall PTR, 1995.
- [11] Rabiner, L. R., and R. W. Schafer, *Digital Processing of Speech Signals*, Englewood Cliffs, NJ: Prentice-Hall, 1978.
- [12] Crochiere, R. E., and L. R. Rabiner, *Multirate Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [13] Dudgeon, D. E., and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [14] Vidyasagar, M., *Nonlinear Systems Analysis*, 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1993.
- [15] Nikias, C. L., and A. P. Petropulu, *Higher-Order Spectra Analysis: A Nonlinear Signal Processing Framework*, Englewood Cliffs, NJ: Prentice Hall PTR, 1993.
- [16] Wornell, G. W., *Signal Processing with Fractals: A Wavelet-Based Approach*, Upper Saddle River, NJ: Prentice Hall PTR, 1996.
- [17] Leon-Garcia, A., *Probability and Random Processes for Electrical Engineering*, 2nd ed., Reading, MA: Addison-Wesley, 1994.
- [18] Helstrom, C. W., *Probability and Stochastic Processes for Engineers*, 2nd ed., New York: Macmillan, 1991.
- [19] Yates, R. D., and D. J. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers*, 2nd ed., Hoboken, NJ: John Wiley & Sons, 2005.
- [20] Hsu, H. P., *Schaum's Outline of Theory and Problems of Probability, Random Variables, and Random Processes*, New York: McGraw-Hill, 1997.
- [21] Poor, H. V., *An Introduction to Signal Detection and Estimation*, 2nd ed., New York: Springer-Verlag, 1994.
- [22] Anastassiou, D., "Genomic signal processing," *IEEE Signal Processing Magazine*, Vol. 18, 2001, pp. 8–20.
- [23] Vaidyanathan, P. P., and Y. Byang-Jun, "The role of signal-processing concepts in genomics and proteomics," *J. Franklin Inst.*, Vol. 341, 2004, pp. 111–135.
- [24] Vaidyanathan, P. P., "Genomics and proteomics: a signal processor's tour," *IEEE Circuits and Systems Magazine*, Vol. 4, 2004, pp. 6–29.
- [25] Watson, J. D., "The human genome project: past, present, and future," *Science*, Vol. 248, Apr. 6, 1990, pp. 44–49.
- [26] Voss, R. F., "Evolution of long-range fractal correlations and 1/f noise in DNA base sequences," *Phys. Rev. Lett.*, Vol. 68, Jun. 22, 1992, pp. 3805–3808.
- [27] Kay, S. M., *Fundamentals of Statistical Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall PTR, 1993.
- [28] Ives, J. T., R. F. Gesteland, and T. G. Stockham, Jr., "An automated film reader for DNA sequencing based on homomorphic deconvolution," *IEEE Trans. on Biomedical Engineering*, Vol. 41, 1994, pp. 509–519.
- [29] Oppenheim, A. V., R. W. Schafer, and T. G. Stockham, Jr., "Nonlinear filtering of multiplied and convolved signals," *IEEE Proc.*, Vol. 56, 1968, pp. 1264–1291.
- [30] Berno, A. J., "A graph theoretic approach to the analysis of DNA sequencing data," *Genome Res.*, Vol. 6, Feb. 1996, pp. 80–91.

- [31] Zhang, X.-P., and D. Allison, "Iterative deconvolution for automatic base calling of the DNA electrophoresis time series," *Workshop on Genomic Signal Processing and Statistics*, Raleigh, NC, Oct. 2002.
- [32] Nelson, D. O., *Improving DNA Sequencing Accuracy and Throughput*, New York: Springer, 1996.
- [33] Giddings, M. C., et al., "An adaptive, object oriented strategy for base calling in DNA sequence analysis," *Nucleic Acids Res.*, Vol. 21, Sept. 25, 1993, pp. 4530–4540.
- [34] Weian, H., et al., "Filter matrix estimation in automated DNA sequencing," *IEEE Trans. on Biomedical Engineering*, Vol. 45, 1998, pp. 422–428.
- [35] Li, L., "DNA Sequencing and Parametric Deconvolution," *Statistica Sinica*, Vol. 12, 2002, pp. 179–202.
- [36] Haan, N. M., and S. J. Godsill, "Modelling electropherogram data for DNA sequencing using variable dimension MCMC," *Proc. 11th IEEE Signal Processing Workshop on Statistical Signal Processing*, Vol. 6, 2001, pp. 3542–3545.
- [37] Haan, N. M., and S. J. Godsill, "A time-varying model for DNA sequencing data," *Proc. 2000 IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 2000, pp. 245–248.
- [38] Boufounos, P., S. El-Difrawy, and D. Ehrlich, "Basecalling using hidden Markov models," *J. Franklin Inst.*, Vol. 341, 2004, pp. 23–36.
- [39] Andrade-Cetto, L., and E. S. Manolakos, "A graphical model formulation of the DNA base-calling problem," 2005, pp. 369–374.
- [40] Ewing, B., and P. Green, "Base-calling of automated sequencer traces using phred. II. Error probabilities," *Genome Res.*, Vol. 8, Mar. 1998, pp. 186–194.
- [41] Ewing, B., et al., "Base-calling of automated sequencer traces using phred. I. Accuracy assessment," *Genome Res.*, Vol. 8, Mar. 1998, pp. 175–185.
- [42] Walther, D., G. Bartha, and M. Morris, "Basecalling with LifeTrace," *Genome Res.*, Vol. 11, May 2001, pp. 875–888.
- [43] Peng, C. K., et al., "Long-range correlations in nucleotide sequences," *Nature*, Vol. 356, 1992, pp. 168–170.
- [44] de Sousa Vieira, M., "Statistics of DNA sequences: A low-frequency analysis," *Phys. Rev. E*, Vol. 60, 1999, p. 5932.
- [45] Li, W., "The study of correlation structures of DNA sequences: a critical review," *Computers & Chemistry*, Vol. 21, 1997, pp. 257–271.
- [46] Li W., and D. Holste, "Universal $1/f$ noise, crossovers of scaling exponents, and chromosome-specific patterns of guanine-cytosine content in DNA sequences of the human genome," *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, Vol. 71, Apr. 2005, p. 041910.
- [47] Li, W., "Expansion-modification systems: A model for spatial $1/f$ spectra," *Phys. Rev. A*, Vol. 43, May 15, 1991, pp. 5240–5260.
- [48] Tiwari, S., S. Ramachandran, A. Bhattacharya, S. Bhattacharya, and R. Ramaswamy, "Prediction of probable genes by Fourier analysis of genomic sequences," *Comput. Appl. Biosci.*, Vol. 13, June 1997, pp. 263–270.
- [49] Yin, C., and S. S. Yau, "A Fourier characteristic of coding sequences: origins and a non-Fourier approximation," *J. Comput. Biol.*, Vol. 12, Nov. 2005, pp. 1153–1165.
- [50] Trifonov, E. N., and J. L. Sussman, "The pitch of chromatin DNA is reflected in its nucleotide sequence," *Proc. Nat. Acad. Sci. USA*, Vol. 77, 1980, pp. 3816–3820.
- [51] Fickett, J. W., "The gene identification problem: An overview for developers," *Computers & Chemistry*, Vol. 20, 1996, pp. 103–118.
- [52] Vaidyanathan, P. P., and Y. Byang-Jun, "Digital filters for gene prediction applications," *Conf. Record 36th Asilomar Conf. Signals, Systems, and Computers*, Vol. 1, 2002, pp. 306–310.
- [53] Vaidyanathan, P. P., and Y. Byang-Jun, "Gene and exon prediction using allpass-based filters," *Workshop on Genomic Signal Processing and Statistics*, Raleigh, NC, Oct. 2002.

- [54] Berger, J. A., S. K. Mitra, and J. Astola, "Power spectrum analysis for DNA sequences," *Proc. 7th Intl. Symp. Signal Processing and Its Applications*, Vol. 2, 2003, pp. 29–32.
- [55] Fox, T., and A. Carreira, "A Digital Signal Processing Method for Gene Prediction with Improved Noise Suppression," *EURASIP Journal on Applied Signal Processing*, 2004, pp. 108–114.
- [56] Byung-Jun, Y., and P. P. Vaidyanathan, "Identification of CpG islands using a bank of IIR lowpass filters [DNA sequence detection]," *IEEE 11th Digital Signal Processing Workshop*, 2004, and *3rd IEEE Signal Processing Education Workshop*, 2004, pp. 315–319.
- [57] Henderson, J., S. Salzberg, and K. H. Fasman, "Finding genes in DNA with a Hidden Markov Model," *J. Comput. Biol.*, Vol. 4, Summer 1997, pp. 127–141.
- [58] Krogh, A., I. S. Mian, and D. Haussler, "A hidden Markov model that finds genes in *E. coli* DNA," *Nucleic Acids Res.*, Vol. 22, Nov. 11, 1994, pp. 4768–4778.
- [59] Lomsadze, A., et al., "Gene identification in novel eukaryotic genomes by self-training algorithm," *Nucleic Acids Res.*, Vol. 33, 2005, pp. 6494–6506.
- [60] Cosic, I., "Macromolecular bioactivity: is it resonant interaction between macromolecules? Theory and applications," *IEEE Trans. on Biomedical Engineering*, Vol. 41, 1994, pp. 1101–1114.
- [61] Ramachandran, P., A. Antoniou, and P. P. Vaidyanathan, "Identification and location of hot spots in proteins using the short-time discrete Fourier transform," *Conf. Record 38th Asilomar Conf. Signals, Systems, and Computers*, Vol. 2, 2004, pp. 1656–1660.
- [62] Pirogova, E., et al., "Investigation of the structural and functional relationships of oncogene proteins," *IEEE Proc.*, Vol. 90, 2002, pp. 1859–1867.
- [63] Cosic, I., "Analysis of HIV proteins using DSP techniques," *Proc. 23rd Ann. Intl. Conf. IEEE Engineering in Medicine and Biology Society*, Vol. 3, 2001, pp. 2886–2889.
- [64] Ideker, T., L. R. Winslow, and A. D. Lauffenburger, "Bioengineering and systems biology," *Ann. Biomed. Engin.*, Vol. 34, Feb. 2006, pp. 257–264.
- [64a] Hanahan, D., and R. A. Weinberg, "The hallmarks of cancer," *Cell*, Vol. 100, No. 1, 2000, pp. 57–70.
- [65] Ideker, T., and D. Lauffenburger, "Building with a scaffold: emerging strategies for high-to low-level cellular modeling," *Trends Biotechnol.*, Vol. 21, June 2003, pp. 255–262.
- [66] Simpson, M. L., et al., "Engineering in the biological substrate: information processing in genetic circuits," *IEEE Proc.*, Vol. 92, 2004, pp. 848–863.
- [67] Rao, C. V., D. M. Wolf, and A. P. Arkin, "Control, exploitation and tolerance of intracellular noise," *Nature*, Vol. 420, Nov. 14, 2002, pp. 231–237.
- [68] Kierzek, A. M., J. Zaim, and P. Zielenkiewicz, "The effect of transcription and translation initiation frequencies on the stochastic fluctuations in prokaryotic gene expression," *J. Biol. Chem.*, Vol. 276, Mar. 16, 2001, pp. 8165–8172.
- [69] Thattai, M., and A. van Oudenaarden, "Intrinsic noise in gene regulatory networks," *Proc. Natl. Acad. Sci. USA*, Vol. 98, July 17, 2001, pp. 8614–8619.
- [70] Simpson, M. L., C. D. Cox, and G. S. Saylor, "Frequency domain analysis of noise in autoregulated gene circuits," *Proc. Natl. Acad. Sci. USA*, Vol. 100, Apr. 15, 2003, pp. 4551–4556.
- [71] Austin, D. W., et al., "Gene network shaping of inherent noise spectra," *Nature*, Vol. 439, Feb. 2, 2006, pp. 608–611.
- [72] Rosenfeld, N., et al., "Gene regulation at the single-cell level," *Science*, Vol. 307, 2005, pp. 1962–1965.
- [73] Pedraza, J. M., and A. van Oudenaarden, "Noise propagation in gene networks," *Science*, Vol. 307, Mar. 25, 2005, pp. 1965–1969.
- [74] Isaacs, F. J., W. J. Blake, and J. J. Collins, "Molecular biology: Signal processing in single cells," *Science*, Vol. 307, Mar. 25, 2005, pp. 1886–1888.
- [75] Newman, J. R., et al., "Single-cell proteomic analysis of *S. cerevisiae* reveals the architecture of biological noise," *Nature*, Vol. 441, June 15, 2006, pp. 840–846.

- [76] Fodor, S. P., et al., "Multiplexed biochemical assays with biological chips," *Nature*, Vol. 364, Aug. 5, 1993, pp. 555–556.
- [77] Schena, M., et al., "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, Vol. 270, Oct. 20, 1995, pp. 467–470.
- [78] Agrawal, H., "Extreme self-organization in networks constructed from gene expression data," *Phys. Rev. Lett.*, Vol. 89, Dec. 23, 2002, p. 268702.
- [79] Rung, J., et al., "Building and analysing genome-wide gene disruption networks," *Bioinformatics*, Vol. 18, Suppl. 2, 2002, pp. S202–S210.
- [80] Tamayo, P., et al., "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation," *Proc. Natl. Acad. Sci. USA*, Vol. 96, Mar. 16, 1999, pp. 2907–2912.
- [81] Alter, O., P. O. Brown, and D. Botstein, "Singular value decomposition for genome-wide expression data processing and modeling," *Proc. Natl. Acad. Sci. USA*, Vol. 97, 2000, pp. 10101–10106.
- [82] Alter, O., P. O. Brown, and D. Botstein, "Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms," *Proc. Natl. Acad. Sci. USA*, Vol. 100, 2003, pp. 3351–3356.
- [83] Furusawa, C., and K. Kaneko, "Zipf's law in gene expression," *Phys. Rev. Lett.*, Vol. 90, Feb. 28, 2003, p. 088102.
- [84] Spellman, P. T., et al., "Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization," *Mol. Biol. Cell.*, Vol. 9, 1998, pp. 3273–3297.
- [85] Butte, A. J., et al., "Comparing the similarity of time-series gene expression using signal processing metrics," *J. Biomed. Inform.*, Vol. 34, Dec. 2001, pp. 396–405.
- [86] Deprettere, E. F., *SVD and Signal Processing: Algorithms, Applications, and Architectures*. European Association for Signal Processing, and Institute of Electrical and Electronics Engineers, Region 8, Amsterdam/New York: North-Holland/Elsevier Science Pub. Co., 1988.
- [87] Golub, G. H., and C. F. Van Loan, *Matrix Computations*, 3rd ed., Baltimore, MD: Johns Hopkins Univ. Press, 1996.
- [88] Janes, K. A., et al., "Cue-signal-response analysis of TNF-induced apoptosis by partial least squares regression of dynamic multivariate data," *J. Comput. Biol.*, Vol. 11, 2004, pp. 544–561.
- [89] Janes, K. A., et al., "A systems model of signaling identifies a molecular basis set for cytokine-induced apoptosis," *Science*, Vol. 310, Dec. 9, 2005, pp. 1646–1653.
- [90] Samoilov, M., S. Plyasunov, and A. P. Arkin, "Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations," *Proc. Natl. Acad. Sci. USA*, Vol. 102, 2005, pp. 2310–2315.
- [91] Korobkova, E., et al., "From molecular noise to behavioural variability in a single bacterium," *Nature*, Vol. 428, Apr. 1, 2004, pp. 574–578.
- [92] Korobkova, E. A., et al., "Hidden stochastic nature of a single bacterial motor," *Phys. Rev. Lett.*, Vol. 96, Feb. 10, 2006, p. 058105.
- [93] Vilar, J. M., R. Jansen, and C. Sander, "Signal processing in the TGF-beta superfamily ligand-receptor network," *PLoS Comput. Biol.*, Vol. 2, Jan. 2006, p. e3.
- [94] Gaudet, S., et al., "A compendium of signals and responses triggered by prodeath and pro-survival cytokines," *Mol. Cell Proteomics*, Vol. 4, Oct. 2005, pp. 1569–1590.
- [95] Goss, P. J., and J. Peccoud, "Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets," *Proc. Natl. Acad. Sci. USA*, Vol. 95, June 9, 1998, pp. 6750–6755.
- [96] Said, M. R., A. V. Oppenheim, and D. A. Lauffenburger, "Modeling cellular signal processing using interacting Markov chains," *Proc. Int. Conf. on Acoustics, Speech, Signal Processing (ICASSP-2003)*, April 2003.

- [97] Michaelis, L., and M. Menten, "Die Kinetik der Invertinwirkung," *Biochem.*, Vol. Z. , 1913, pp. 333–369.
- [98] Gu, X., and N. C. Spitzer, "Distinct aspects of neuronal differentiation encoded by frequency of spontaneous Ca^{2+} transients," *Nature*, Vol. 375, June 29, 1995, pp. 784–787.
- [99] Hajnoczky, G., et al., "Decoding of cytosolic calcium oscillations in the mitochondria," *Cell*, Vol. 82, Aug. 11, 1995, pp. 415–424.
- [100] Brady, K. D., et al., "Alterations in the frequency and shape of Ca^{2+} fluctuations in GH4C1 cells induced by thyrotropin-releasing hormone and Bay K 8644," *Biochem. J.*, Vol. 306, Part 2, Mar. 1, 1995, pp. 399–406.
- [101] Wagner, K. A., et al., "Mechanism of spontaneous intracellular calcium fluctuations in single GH4C1 rat pituitary cells," *Biochem. J.*, Vol. 292, Part 1, May 15, 1993, pp. 175–182.
- [102] Costantin, J. L., and A. C. Charles, "Modulation of Ca^{2+} signaling by K^{+} channels in a hypothalamic neuronal cell line (GT1-1)," *J. Neurophysiol.*, Vol. 85, Jan. 2001, pp. 295–304.

Signal Processing Methods for Mass Spectrometry

Peter Monchamp, Lucio Andrade-Cetto,
Jane Y. Zhang, and Robert Henson

4.1 Introduction

With the advent of important advances in instrumentation, researchers nowadays can perform large-scale experiments on biological data. They aim to understand biological processes and functions by measuring data at the molecular and cellular level. The large number of required experiments compared with the limited number of measurable events gives signals that are frequently immersed in noise and have poor quality. For example, high-throughput DNA sequencing appeared in the late 1990s at the peak of the Human Genome Project [1, 2] and pushed the rate of data acquisition to its limits. Inferring the DNA sequence from four time traces (base-calling) was significantly improved by preprocessing the signal. Gel electropherograms and later capillary electrophoresis were enhanced with deconvolution methods, background subtracting, signal decorrelation, normalization, and other methods well known at the time by the signal processing community [3]. Microarray technologies, which measure gene expression at the cell level by testing mRNA, also required algorithms borrowed from signal processing for normalization and smoothing [4]. In this chapter we review the signal processing techniques that are used with mass-spectrometry signals. Other new technologies now being developed, such as liquid chromatography mass spectrometry (LC-MS) and tissue microarrays, will also require preprocessing to improve the data.

The use of mass spectrometry (MS) to diagnosis disease by identifying the proteins in biological samples has been gaining interest in recent years [5]. As a first step, biological fluids, such as serum, are analyzed for protein patterns without identifying the underlying proteins [6]. Differences in protein patterns between diseased and healthy patients can occur because of differences in the expressed proteins. Further analysis identifies the proteins responsible for the disease as biomarkers [7]. In this case, biomarkers can be one or more proteins that, when detected and measured, indicate the presence of a specific disease. Clinicians can use these biomarkers for diagnosis and prognosis, while pharmaceutical researchers can investigate biomarkers as possible drug targets or to understand biochemical pathways.

4.1.1 Data Acquisition Methods

MS is an analytical technique for identifying molecules using information about their mass or the mass of their fragments. Any molecule that can be ionized into the gas phase can have its mass determined by a mass spectrometer. An ion source vaporizes molecules into the gas phase and converts them into ions. The gas phase ions are accelerated through an electric field and separated by their mass (m) and charge (z). Finally, the separated ions are detected and measured by an electron multiplier. The MS data is plotted as a spectrum with m/z values on the x -axis and ion intensity on the y -axis.

There are four common techniques for ionizing biological molecules. Electron Ionization (EI) is the most common ionization technique. It works well for small molecules that are easily vaporized into the gas phase. With thermally sensitive molecules, EI causes extensive fragmentation where you may not observe the parent ion. For large biological molecules with low volatility and thermal instability you need to use other methods of ionization. Soft ionization techniques such as Fast Atom Bombardment (FAB), Electrospray Ionization (ESI), and Matrix-Assisted Laser Desorption Ionization (MALDI) overcome the limitations of EI. Currently, the most common methods for ionizing large biological molecules are Electrospray Ionization Mass Spectrometry (ESI-MS), Matrix-Assisted Laser Desorption Ionization Mass Spectrometry (MALDI-MS) and Surface Enhanced Laser Desorption Ionization Mass Spectrometry (SELDI-MS). These methods can detect high molecular mass, low volatile, and thermally labile compounds such as proteins in biological samples. They all use soft ionization techniques to volatilize the proteins into the gas phase without fragmenting the molecules and to detect them with high sensitivity.

4.1.2 History of Ionization Techniques

The developers of two of the common ionization techniques in mass spectrometry received Nobel Prizes. In 2002, John Fenn (electrospray ionization) and Koichi Tanaka (soft laser desorption ionization) shared half of the Nobel Prize in Chemistry for their development of techniques to analyze biological macromolecules using mass spectrometry. In both cases, the discovered breakthroughs were related to extending the size of biological molecules that could be analyzed to over 10,000 Daltons.

John Fenn developed electrospray ionization (ESI), where proteins in a liquid solvent are sprayed through a nozzle with a strong voltage applied to produce charged droplets. Solvent is then removed from the charged droplets, leaving charged protein ions. With ESI, the ionized proteins are produced with a series of multiple charges. The breakthrough for analysis of large molecules described by John Fenn in 1989 [8] was to add a counterflow of gas to desolvate the droplets and use signal averaging over the multiple ions for a single protein to create a signal that was stronger and more accurate than any of the individual ion signals.

Koichi Tanaka developed soft laser desorption ionization (SLDI), a precursor to MALDI, where proteins are mixed with a matrix material and applied to a metal plate. A laser ionizes and vaporizes the matrix and protein molecules from the plate. The breakthrough described by Koichi Tanaka in 1988 [9] was to use a ma-

trix material of ultra-fine cobalt particles and glycerol with a low-energy nitrogen laser having a wavelength of 337 nm to ionize the proteins. Energy from the laser is selectively absorbed by the matrix, while the proteins tend not to absorb light with a wavelength of 337 nm. Using a combination of laser wavelength and matrix material, large proteins are vaporized and ionized without fragmentation.

4.1.3 Sample Preparation

Electrospray (ESI) does not need any prior sample preparation. Sample molecules in a liquid are separated using liquid chromatography (LC) techniques with the liquid from the end of a chromatography column introduced directly into an ES ionizer.

Samples for MALDI are prepared by mixing a matrix solution with a sample solution and spotting the mixture on a MALDI plate. The plate is allowed to dry while solvents in the mixture evaporate, leaving a crystallized matrix.

SELDI is a similar technique to MALDI. It is a proprietary analysis method from Ciphergen Inc. for selectively separating proteins from a mixture. With SELDI, a biological sample is applied to a surface with an affinity for proteins with different chemical properties. Proteins with an affinity for the surface bond to it, while proteins without an affinity are washed off the surface. A matrix solution is next applied over the sample and allowed to dry and crystallize.

4.1.4 Ionization

With ESI, a stream of liquid is pumped from an LC column through a needle with a very high voltage. The charged liquid is broken into droplets with a nebulizing gas, and then solvent molecules are removed from the sample with a stream of drying gas. By a method that is not clearly understood, charge on the solvent molecules is transferred to the sample molecules with the addition of one or more protons. The remaining sample ions in the gas phase are attracted to the entrance of the MS detector.

After sample preparation, MALDI and SELDI use the same instrument technique. The crystallized mixture is inserted into an ion source with a high vacuum. It is irradiated with a laser. The matrix molecules absorb most of the energy and protect the sample from being fragmented. Matrix molecules desorb from the surface of the plate and vaporize along with the sample molecules. Energy is transferred from the matrix molecules to the sample molecules to help them ionize. Protein molecules are usually ionized by adding a proton (H^+) to the molecular ion (M) to create a singly charged ion $[M+H]^+$, but there may also be some doubly charged proteins $[M+2H]^{2+}$.

4.1.5 Separation of Ions by Mass and Charge

A common method for separating ions with MALDI samples uses a time-of-flight (TOF) tube. Positively charged sample ions formed in the source are repelled by a positively charged anode and accelerated into a mass analyzer by an electric field into a flight tube. The molecules traveling down the flight tube reach the ion detector at different times because of differences in mass and charge. The higher the mass of an

ion, the lower its velocity and the longer it takes to travel down the flight tube to the detector. Ions with twice the charge move twice as fast as ions with the same mass but half the charge.

The time for an ion to reach a detector from the source is given by (4.1), in which $(t - t_0)$ = time of flight for an ion from the source to the detector, M = mass of the ion, e = charge of the ion, E = electric field to accelerate ions into the flight tube, d = length of accelerating region with electric field, L = length of nonaccelerating region without an electric field, and V_0 = potential of the electric field.

$$t - t_0 = \left(\frac{2md}{ze} \right)^{1/2} + L \left(\frac{m}{2zeV_0} \right)^{1/2} \quad 4.1$$

After rearranging (4.1) for m/z , the quadratic relationship between the mass-to-charge ratio and TOF is apparent in (4.2). The constants a and b depend on the instrument, potential applied at the source, electric field, and length of the flight tube.

$$m/z = a(t - t_0)^2 + b \quad 4.2$$

Some MALDI-TOF instruments have an ion mirror that deflects ions with an electric field back down the flight tube. Doubling the flight path of ions increases the resolution between ion peaks [10].

4.1.6 Detection of Ions and Recorded Data

An electron multiplier detects and measures the ions reaching the end of a TOF tube. After an MS instrument is calibrated with compounds of known mass, the constants in the quadratic equation relating time to mass/charge are determined, and the mass/charge of detected ions calculated. The result is a series of data points with mass/charge and relative ion intensity values. A mass spectrum is a plot of mass/charge on the x -axis and relative ion intensity on the y -axis.

For large biomolecules, a MALDI-MS instrument can measure the molecular mass with an accuracy sufficient to identify individual peptides.

4.1.7 Data Preprocessing

Experimental MS data begins with data acquisition, uses preprocessing to correct some of the acquisition problems, and ends with analysis to identify protein molecules. Before analyzing spectra data, you need to preprocess it to remove or minimize problems with the data [11]. Problems with data acquisition can be divided into two areas:

- Flawed experimental design and technique. This area includes samples prepared with different procedures, sample data sets not acquired randomly to minimize systemic errors, and comparing spectra acquired with different instruments. Problems with the experimental process need to be corrected before you can preprocess data and complete the final analysis [12, 13].
- Instrument miscalibration, noise, and variation. The processing methods described in this chapter can minimize problems in this area, but cannot correct

for poorly acquired data from problems in the previous area. Processing techniques cannot overcome problems with inadequate data acquisition technique.

In contrast to the processing methods in this chapter, other classical MS analysis strategies keep information only for the mass of peaks calculated by an instrument. The detected ion intensity of a peak is characterized by determining the centroid of the peak, and then representing it with a single intensity value equal to the peak height and assuming the m/z value at the centroid corresponds to the actual mass. The instrument completes the preprocessing steps using black-box algorithms. The advantage to this approach is that it saves a huge amount of memory. The disadvantage is that important information might be lost due to a defective peak extraction or failed segmentation. Lost information could happen if peaks appear overlapped in the raw spectra and the shape of the peaks is distorted due to a low signal-to-noise ratio. Processing the raw data allows you to improve the results from further analysis of the data.

4.1.8 Example Data

This chapter shows a typical workflow for dealing with protein MS data. The example data are from the Federal Drug Administration–National Cancer Institute (FDA-NCI) Clinical Proteomics Program Databank and was used to identify proteomic patterns for diagnosis of ovarian cancer in serum samples [14]. The data was acquired using Surface-Enhanced Laser Desorption Ionization Time-of-Flight Mass Spectrometry (SELDI-TOF MS) [15].

4.2 Signal Resampling

Signal resampling is the process of calculating a new signal with intensity values at controlled mass/charge (m/z) points where the reassembled signal follows, as much as possible, the original signal. By controlled we mean that the mass/charge points can be less than the original ones (down-sampling), approximately equal (synchronizing), or more than (up-sampling). In mass spectrometry, up-sampling is usually not used.

With high-resolution MS data, the large number of values in a signal can be impractical to work with using computationally intensive algorithms, and they may reach the limits of computer memory. If the sampling rate is higher than the resolution of the instrument, you could have redundant values immersed in noise, or your analysis may not need the data provided with a higher resolution. In both cases, you could remove the extra values.

Another problem is that the number of m/z values and the distance between m/z values may vary between samples analyzed with one instrument or, more likely, with different instruments, making comparison between spectra difficult.

Resampling has several advantages. By resampling you can:

- Reduce the values in a signal to a more manageable number while preserving the information content of the spectra. If the datasets are too large to keep in the available memory, then you need to down-sample to be able to work with

all of the data. You may also want to do this for algorithm design purposes and work with a smaller dataset;

- Take spectra with different m/z vectors and match the scales, creating a consistent m/z vector range. If the samples were taken from different machines, then the values may be slightly different, so you need to resample to get everything on the same scale. Also, comparative algorithms between spectra may need to use the same reference values;
- Fill in missing m/z values. Another issue is that samples may be missing for certain m/z values so you can use resampling to fill in dropped values. This helps when you need to visualize the data. Dropped samples can only be recovered if the original m/z values follow a linear or a quadratic function.

A disadvantage of resampling occurs if you reduce the number of values for visualization and analysis purposes to a size that masks or removes important features of the data.

You want a function that allows you to select a new m/z vector by specifying an m/z range and the number of values. It inputs a raw mass spectrum and outputs a spectrum having the specified number of samples with an m/z spacing that increases linearly within the specified range. The m/z vector can be a linear or a quadratic function.

Also apply an antialias filter to prevent high-frequency noise from folding into lower frequencies. The antialias filter could use a linear-phase Finite Impulse Response (FIR) filter with a least-squares error minimization. The cut-off frequency is set by the largest down-sampling ratio when comparing the same regions in the m/z input and output vectors [16].

4.2.1 Algorithm Explanation and Discussion

Resampling calculates new m/z points with their respective values that best fit to the original raw spectra. The new m/z values should be regularly spaced following a known function $f(x)$. For digital signal processing, this is similar to sample rate conversion where $f(x) = K$. In genomic signal processing, $f(x)$ could be a soft function, so you can have more samples in the areas with a high content of information. For example, TOF signals have a quadratic relationship between mass and charge (4.1), where you would want to have more samples in the low m/z values of the spectra. When looking at different spectra, resample all spectra to the same $f(x)$. This allows you to further compare spectra without having to segment the signals further. Working with low-resolution spectra from different experiments might require you to resample to improve the reproducibility of experiments.

When down-sampling a signal, high-frequency components appear in the down-sampled signal as low-frequency components known in the signal processing community as aliasing. To prevent aliasing, you should figure out the Nyquist frequency ($f_N = f_{\text{Sampling}}/2$) and prefilter the original signal before down-sampling. In the case of high-resolution MS signals, the high-frequency content of the signal is mostly noise. Since the sampling rate may be variable for a single spectrum, the Nyquist frequency is also variable. For practical cases, select the Nyquist frequency

with a value equal to the minimum distance between two contiguous samples of the targeted m/z vector.

4.2.2 Example Demonstrating Down Sampling

In this section, a high-resolution example taken from the FDA-NCI ovarian dataset is used to demonstrate how to resample MS data. Functions from the Bioinformatics Toolbox [17] show the process of converting high-resolution spectra to low-resolution spectra by down-sampling. Load the high-resolution spectra and plot the data.

```
load high_resolution_sample;  
plot(MZ, Y, '.');
```

The first variable MZ is a vector of m/z values, while the second variable Y is a vector of ion intensity values corresponding to each m/z value. See Figure 4.1 for a plot of the raw MS data.

Determine the number of data values in the original spectrum.

```
original_size = numel(MZ)  
original_size =  
355760
```

Down-sample the spectra between 2,000 and 11,000 and reduce the number of data values.

```
[MZD,YD] = msresample(MZ,Y,10000,'Range',[2000 11000]);
```

Plot the resampled spectrum and notice the reduced number of data points. See Figure 4.2 for an example of a spectrum with additional data points removed.

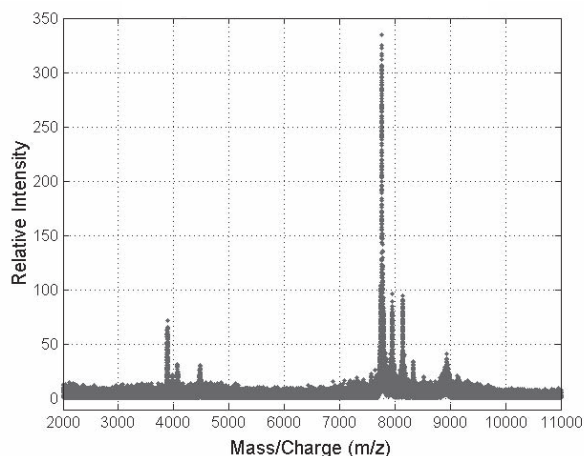


Figure 4.1 Original mass spectrum before resampling.

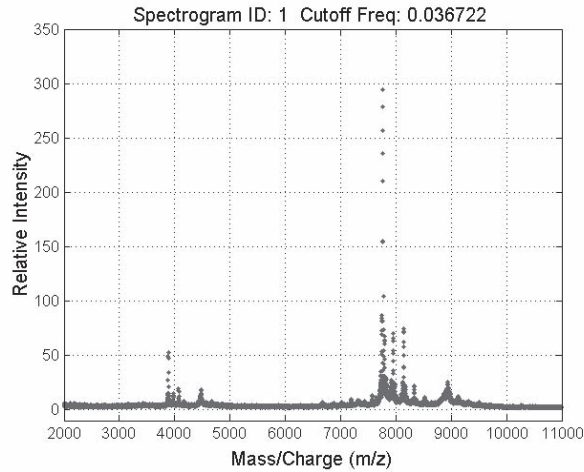


Figure 4.2 Resampled mass spectrum.

Zoom into a section of the spectrum.

```
axis([3875 3895 0 90])
```

See Figure 4.3 comparing a spectrum before and after resampling with the antialiasing filter turned on.

Resample the original spectrum but this time turn off the antialias filter. The down-sampled spectrum shows some noise due to aliasing effects. See Figure 4.4.

```
[MZD,YD] = msresample(MZ,Y,10000,'Range',[2000 11000],  
'Cutoff', 1.0, 'ShowPlot', true);  
axis([3875 3895 0 90])
```

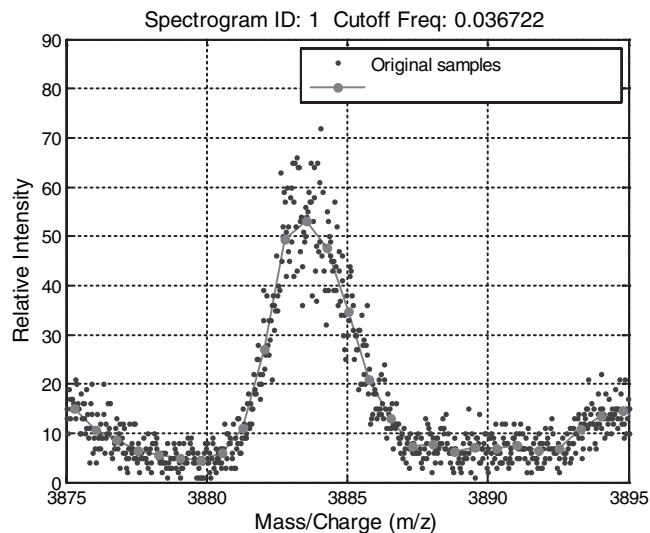


Figure 4.3 Mass spectra with antialias filtering.

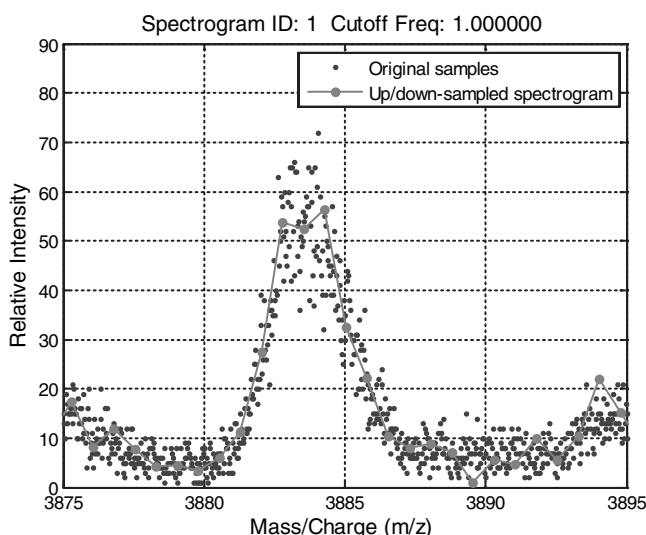


Figure 4.4 Resampled spectrum without antialias filter.

4.3 Correcting the Background

Mass spectrometry data usually shows a varying baseline. Chemical noise in the matrix or ion overloading can cause this variation. Subtracting the baseline makes spectra easier to compare. Use baseline correction:

- With samples that have an obvious offset, drift, or broad low-frequency peaks;
- After you down-sample or with spectra that have consistent m/z ranges;
- Before you correct the calibration, because the noise will affect the results of that step. MALDI and TOF samples are particularly susceptible to noise, although other techniques and more sensitive instruments give cleaner spectra.

One strategy for removing a low-frequency baseline within the high-frequency noise and signal peaks follows three steps: (1) estimate the most likely baseline in a small window, (2) regress the varying baseline to the window points using a spline interpolation and smoothing, and (3) subtract the estimated and regressed baseline from the spectrum. Also, consider band broadening of mass ion peaks by assuming a Gaussian distribution of peaks and plotting the standard deviation across the m/z values, and then use a monotonic smoothing algorithm to subtract the baseline [18].

4.3.1 Algorithm Explanation and Discussion

Estimating the most likely background in every window is the most crucial step. Unfortunately, you cannot observe the true baseline using the minimum values because of the high-frequency signal noise. There are two good approaches to overcome this problem:

- Use a quantile value of the observed sample within the window (see Figure 4.5). This approach is fast, but it has the disadvantage of assuming there are a

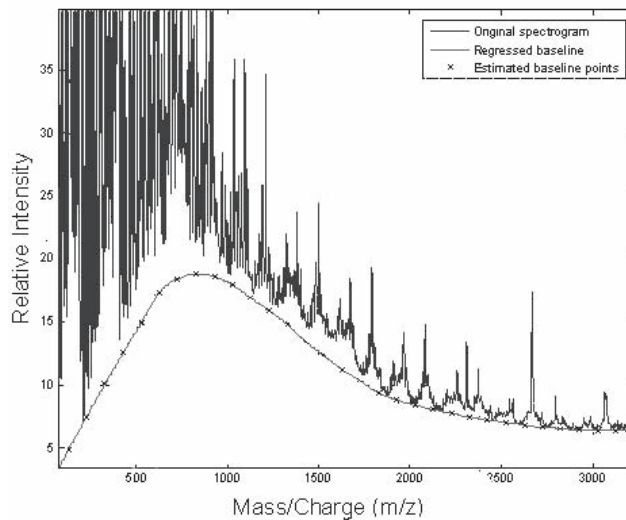


Figure 4.5 Baseline estimated using quantile values.

relatively equal proportion of points that belong to a peak and points that belong to the baseline in the current window. For example, by setting the quantile to 10%, you are assuming that in every window 20% of the points are baseline and 80% are peaks. You can safely underestimate the quantile with the result that your estimated baseline is slightly conservative. On the other hand, if you over estimate the quantile, the proportion of baseline points includes peak values. See Figure 4.5.

- Use a probabilistic model. The second approach improves the result at the cost of computational time. You can assume that the points in every window come from a doubly stochastic model, that the source of each point can be “noise” or “peak,” and that each class has its own distribution. In practice, assuming a uniform Gaussian distribution is relatively safe. Estimating the baseline implies learning the distributions and the class labels for every point, which is an unsupervised clustering problem solved by an Expectation-Maximization estimation [18]. At the end, the mean of the “noise” class turns out to be the best baseline estimate for the window. See Figure 4.6.

How do you select the window size? It should be sufficiently small so that the varying trend of the baseline is not significant, and you can assume that it is constant in your estimation. This makes the estimation approach faster and be more robust. It should be sufficiently large so that you can observe a representative sample of the baseline in the window. In the case of MS signals, the abundance of peaks and resolution of the raw trace varies through a spectrum, so you should allow different windows sizes, depending on the region of the spectrum.

Why use spline interpolation and smoothing to regress the baseline? Some authors have tried to approximate the baseline of a signal to a known function. For example, using a known function is a good strategy with genomic signal preprocessing and DNA sequences, where a combination of exponential and linear curves is sufficient to model the background of DNA chromatograms. This strategy satisfactorily recovers the baseline introduced by gel electrophoresis. When you can de-

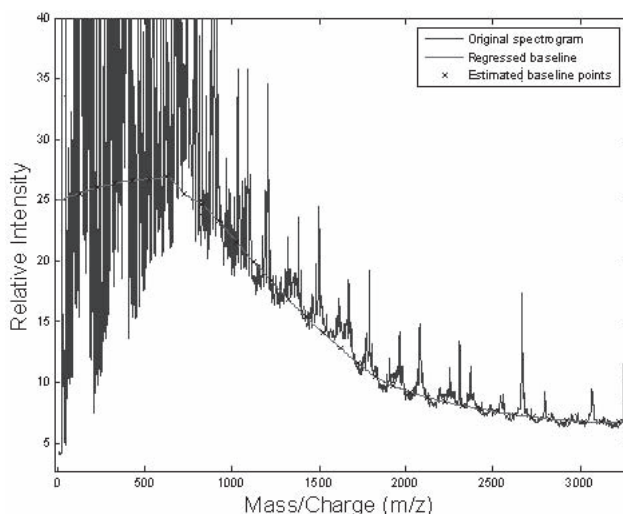


Figure 4.6 Baseline estimation using a probabilistic model.

rive a good model that correlates to the observed data, the function-based regression is more robust. With MS signals, you would have difficulty finding a good model that always correlates to the background, but you could smooth the acquired data points and then interpolate for every m/z value. The smoothing is sufficient to minimize the effect of potential outliers.

4.3.2 Example Demonstrating Baseline Subtraction

In this and the following sections, four low-resolution spectra taken from two different low-resolution ovarian cancer FDA-NCI ovarian datasets are used to demonstrate MS preprocessing tasks. These spectra were generated using the WCX2 protein-binding chip, two with manual sample handling and two with a robotic sample dispenser and processor. Functions from the Bioinformatics Toolbox show the process for correcting a baseline. Load a set of low-resolution spectra and plot the data for the second spectra.

```
load low_resolution_sample;
plot(MZ,Y(:,2));
```

MZ is the mass/charge vector, while Y is a matrix, with the ion intensities for each sample in separate columns. See Figure 4.7 for a plot of the raw MS data.

Adjust the baseline for a set of spectra by selecting a window of 500 points and assuming 20% of the points in a window are baseline, and plot the second spectrum with the estimated baseline subtracted.

```
YB = msbackadj(MZ,Y,'WindowSize',500,'Quantile',0.20);
plot(MZ, YB(:,2));
```

See Figure 4.8 for an example of a spectrum with the baseline subtracted from the raw spectrum.

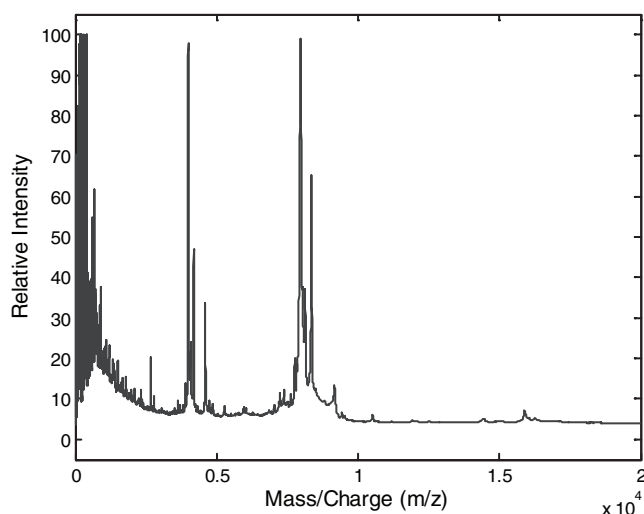


Figure 4.7 Low-resolution mass spectrum example.

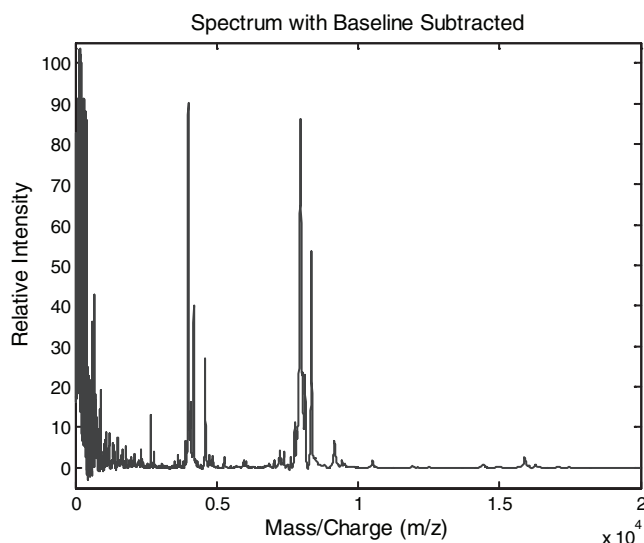


Figure 4.8 Spectrum with estimated baseline subtracted.

4.4 Aligning Mass/Charge Values

Errors in calibration or limitations of a mass spectrometer can lead to variations between the observed m/z vector and the true TOF of the ions. Therefore, systematic shifts appear in repeated experiments and two identical proteins acquired in different spectra can have different m/z values. A single instrument or using different instruments can cause these systematic errors. Although the high-throughput detector used in MS can generate numerous spectra per patient, undesirable variation may get introduced in the MS data due to the nonlinearity in the detector response, ionization suppression, minor changes in the mobile phase composition, and inter-

action between analytes. Additionally, the resolution of the peaks usually changes for different experiments and also varies towards the end of the spectrum.

Adjust the m/z values when:

- A known profile of peaks is expected in a spectrum. You may have known compounds in a biological sample that you know should align.
- Your samples are “spiked” with known compounds (internal standards) to aid calibration.
- External calibration standards analyzed with samples show variation.

Resample and correct the baseline for your raw spectra before trying to align m/z values. One advantage of working with the raw data, and not with peak information, is that the alignment algorithm is less prone to fail due to a defective peak extraction algorithm. When analyzing MALDI-TOF data, you may have information about m/z values for known calibration standards or contaminants. A preprocessing function should use a set of m/z values where you expect reference peaks to appear, and it should allow you to define a set of relative weights which the aligning algorithm can use to emphasize peaks with a small area.

One method aligns a raw mass spectrum by scaling and shifting the m/z scale so that the cross-correlation between the raw mass spectrum and a synthetic spectrum is maximized. Build a synthetic spectrum with Gaussian pulses centered at the masses specified by the reference peaks. Once a new m/z scale is determined, calculate a new spectrum by piecewise cubic interpolating and shifting the new spectrum from the original m/z vector. This method preserves the shape of the peaks.

4.4.1 Algorithm Explanation and Discussion

A smooth function warps the signals by resampling the spectra. The smooth function can be any higher-order polynomial. Since most of the observed shifts in the MS data are due to the difficulty of achieving a consistent calibration of the TOF to mass between experiments (4.1), the function `msalign` in the Bioinformatics Toolbox uses a second-order warp function. Other authors [19] have proposed using cubic splines for datasets in which the dominant shift anomalies are not due to the former quadratic relation.

The alignment algorithm builds a synthetic signal with two or more peaks represented by a Gaussian kernel. The m/z values of the synthetic signal (the location of the Gaussian peaks) are shifted and scaled until the cross-correlation between the raw mass spectrum and the synthetic signal reaches its maximum value. In this case, shifting and scaling represent the two degrees of freedom needed in the smooth warping function. For higher-order warp functions, you would need to identify more parameters. The user is responsible for selecting the approximate location of the reference peaks expected to appear in the spectra.

When multiple spectra are aligned, the previous algorithm is repeated for each one. The estimation of the warping function for every spectrum can be distributed over a cluster of computers since these computations are data independent, therefore achieving linear speedup of the computations. The algorithm then selects the ultimate locations of the reference peaks based on the computed warping functions such that the sum of the squared shifts for the reference peaks is minimized. A

substantial difference between this alignment approach and other published approaches [20] is that this approach infers the warping function from the raw data and not from a list of peaks.

Setting the width of the Gaussian pulses has a twofold purpose. On one side, pulses should be narrow enough so that close peaks in the spectra are not included with the reference peaks. On the other side, pulses should be wide enough so that the algorithm captures a peak that is off the expected site. Tuning the spread of the Gaussian pulses controls a tradeoff between robustness (wider pulses) and precision (narrower pulses). However, pulse width is unrelated to the shape of the observed peaks in the spectrum. The algorithm allows you to give spectrum-dependent widths and weights to every reference peak. You may want to set different widths for Gaussian pulses since the typical spectrum resolution changes along the m/z range. Peak weights are used to emphasize peaks whose intensity is small but that provide a consistent m/z value and appear with good resolution in most of the spectra.

The algorithm searches over a two-dimensional grid of possible shifts and scales for the m/z vector using a multiresolution exhaustive grid search. This approach does not guarantee you will find a global maxima. However, since misalignments of peaks generally are systematic and small, the algorithm adjusts the m/z values while preserving its robustness for noisy datasets. You can improve this technique by using a better optimization method instead of an exhaustive grid search. For example, you could apply genetic algorithms, which considerably speed up the estimation of the warping functions.

4.4.2 Example Demonstrating Aligning Mass/Charge Values

Plot four low-resolution spectra with the baseline corrected, and then zoom into a few ion peaks to show the misalignment of m/z values between spectra.

```
plot(MZ,Y);
```

See Figure 4.9 for a plot of four misaligned mass spectra.

Enter the location and weight of the reference peaks.

```
P = [3991.4 4598 7964 9160];
```

```
W = [60 100 60 100];
```

Use a heat map to observe the alignment of peaks in the original spectrum. See Figure 4.10.

```
msheatmap(MZ,YB,'Markers',P,'Limit',[3000 10000]),  
title('Before Alignment')
```

Align the set of baseline-subtracted spectra to the reference peaks given.

```
YA = msalign(MZ,YB,P,'Weights',W);
```

After applying the alignment algorithm, you can observe improvements in peak alignment between spectra based on peak height. See Figure 4.11.

```
msheatmap(MZ,YA,'markers',P,'limit',[3000 10000])
```

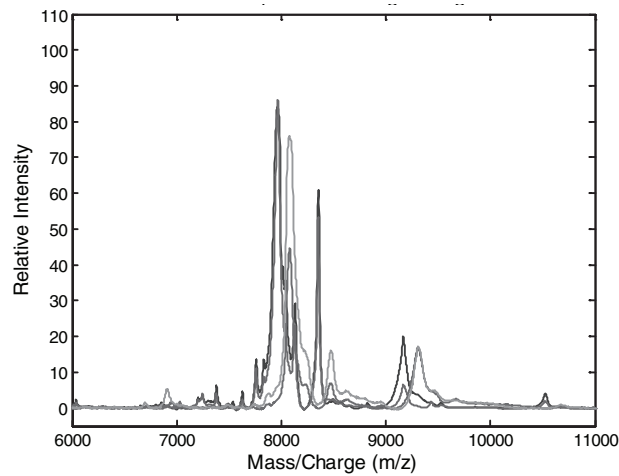


Figure 4.9 Four low-resolution mass spectra showing misalignment.

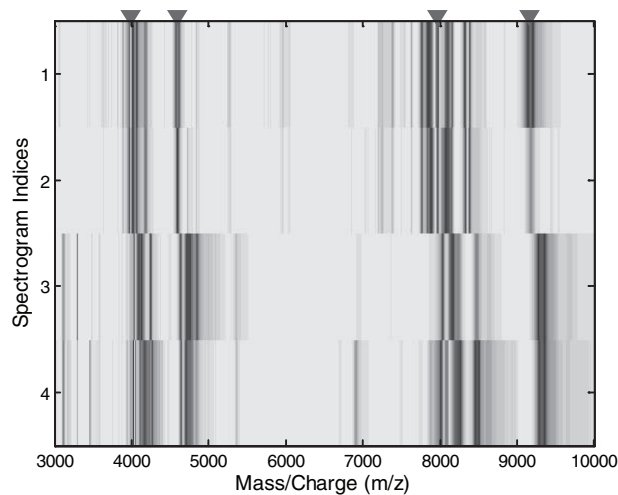


Figure 4.10 Heat map showing misalignment.

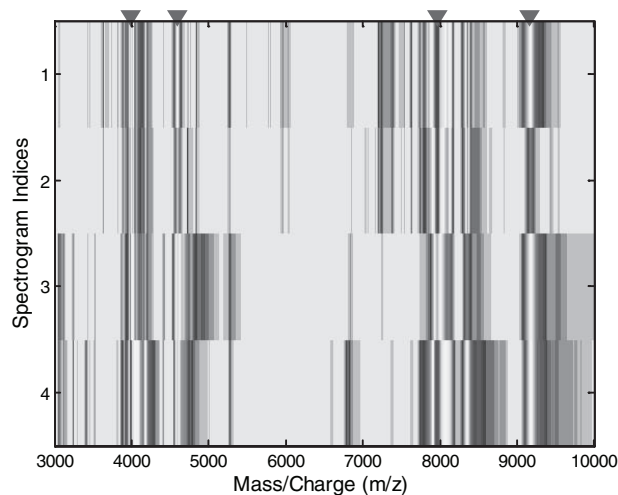


Figure 4.11 Heat map showing alignment corrected.

4.5 Normalizing Relative Intensity

Repeated experiments commonly have systematic differences in the total amount of desorbed and ionized proteins. Sample sizes may be different, sample preparation may be different with different technicians, there could be ion saturation, or the sensitivity of the instrument may change. The result is a variation in the amplitude of ion intensities.

To compensate for systematic differences, you could normalize the relative intensities of the spectra to the average area under the spectra curves or the height of a selected peak. This type of normalization has been used in experiments looking for differences in expressed proteins, but there is an assumption about the samples. The assumption is that the amount of proteins whose expression changes is much less than the amount of total proteins in a sample. This may not always be the case.

A second, more robust normalization method uses the area or height of an internal standard. An internal standard is a compound with a known mass and with the same amount of compound added to each sample. Differences in the area of an internal standard are proportional to the differences in area for the proteins in a sample.

Normalize your samples

- After subtracting the baseline and correcting miscalibration by adjusting the m/z values;
- After subtracting the low m/z values with ion intensity values having considerable noise;
- When the samples are “spiked” with known compounds (internal standards).

You can normalize a group of mass spectra by setting the area under each curve to the group median or to the percentage of height of a selected peak, or you can normalize samples with a constant amount of “spiked” internal standard with the area of the standard peak [21–24].

4.5.1 Example Demonstrating Intensity Normalization

Plot the low-resolution spectra after correcting for baseline variation and miscalibration. See Figure 4.12.

```
plot(MZ, YA)
```

One of many methods to normalize the intensity values of spectra is to rescale the maximum intensity of every signal to a certain value. For example, you could select the highest peak in a sample and normalize all spectra to 100% of that peak. It is also possible to ignore problematic regions. For example, in biological samples you might want to ignore the low-mass region ($m/z < 1000$ Daltons). Choose a cutoff value that eliminates the large amount of noise at lower m/z values but does not remove any proteins of interest.

```
YN1 = msnorm(MZ,YA,'Quantile',1,'Limits',[1000 inf],'MAX',100);  
plot(MZ,YN1);
```

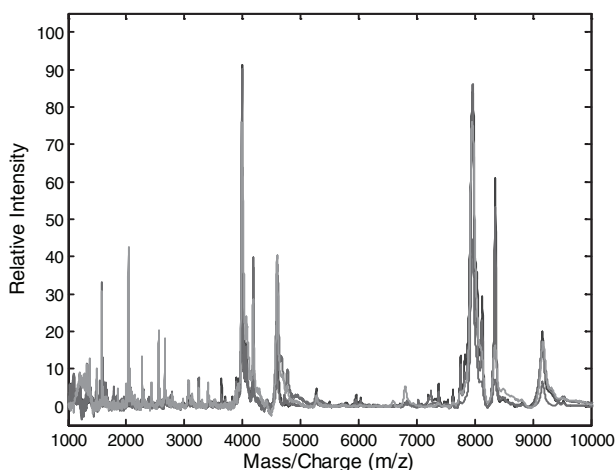


Figure 4.12 Four spectra with baseline and calibration corrected.

See Figure 4.13 for an example of four spectra normalized to the highest ion peak from one of the spectra.

The `msnorm` can also normalize using the area under the curves and then rescaling the spectra having relative intensities below 100.

```
YN2 = msnorm(MZ,YA,'LIMITS',[1000 inf],'MAX',100);
plot (MZ, YN2)
```

See Figure 4.14 for an example of four spectra normalized to the mean area from the four spectra.

You can also use the peak height or area of an internal standard to normalize the spectra for comparison. For example, if the peak at 9164 is an internal standard, you could normalize a set of spectra based only on the mean area of this peak.

```
plot(MZ, YA); axis([8500 10000 -5 105]);
```

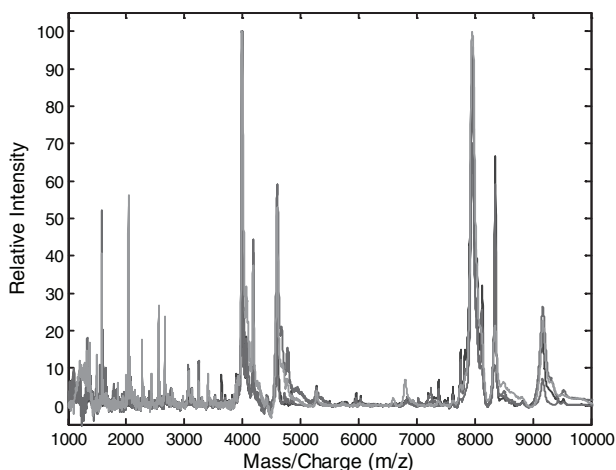


Figure 4.13 Four spectra normalized to the highest ion peak.

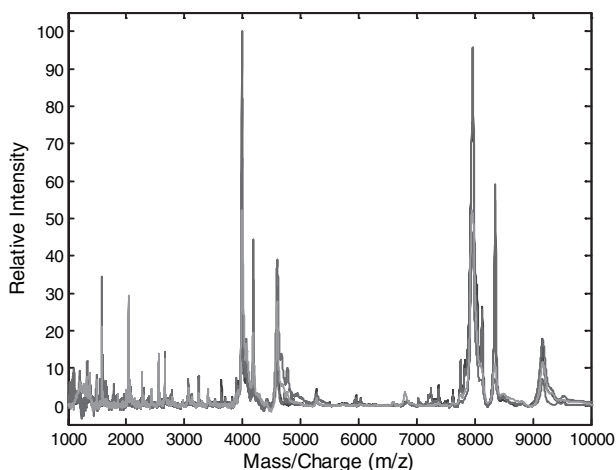


Figure 4.14 Four spectra normalized to the mean area under the curves.

View the variation in area (ion intensity) of the peak at 9164 m/z in Figure 4.15, which shows an ion peak from four spectra with the same amount of compound but different areas.

Normalize the area of the peak at 9164 to 40% of its height. By setting the quantile to 0.8, `msnorm` uses the highest 80% of values in the selected window to normalize the peak. This eliminates normalization errors from smaller peaks riding on the edge of the selected peak.

```
YN3 = msnorm(MZ, YA, 'limits', [9000 9300], 'quantile', [0.8 1],
    'MAX', 40);
plot(MZ, YN3); axis([7000 10000 -5 105]);
```

See Figure 4.16 for an example of an ion peak in four spectra normalized to have the same area.

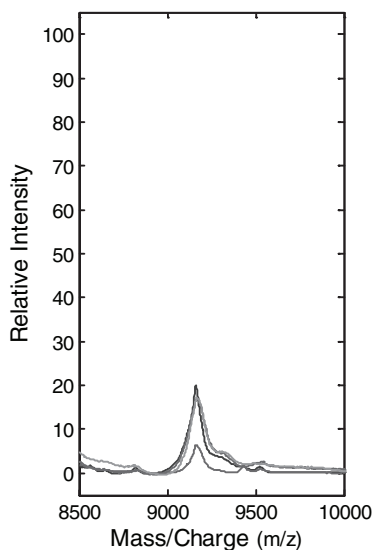


Figure 4.15 Internal standard with unequal areas.

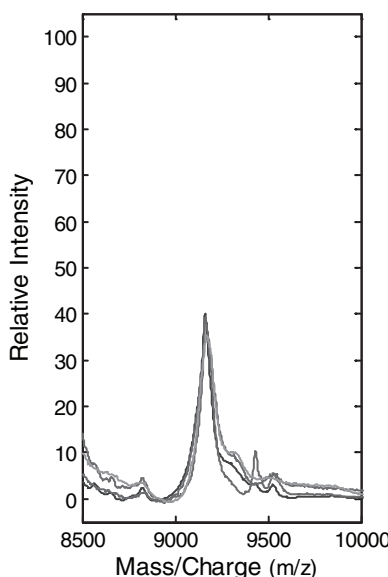


Figure 4.16 Spectrum normalized to an internal standard.

4.6 Smoothing Noise

Mass spectra usually contain a mixture of noise and signal. Some applications require you to filter the noise or smooth the spectra in order to improve the validity and precision of the observed m/z values of the peaks in the spectra. For the same reason, smoothing also improves peak detection algorithms. Noise reduction can be achieved either by filtering or by smoothing. This section reviews the smoothing techniques and explains why these are the preferred techniques to enhance the signal over conventional filtering.

Filtering is a noise reduction technique that selectively reduces the power content of specific signal frequencies. There are two families of filters, IIR and FIR, that you can apply efficiently to the signal. You need to design a filter in terms of its frequency characteristics. For this you can analyze the frequency spectrum of the signal and try to identify the frequencies of the undesired content and choose the best cutoff frequency for your filter. In the case of mass spectra, you need a low-pass filter since the low-frequency noise of the signal and baseline has already been corrected. The length of the filter depends on the degree of frequency selectiveness you want.

Smoothing (also known as polynomial filtering) is an alternative for noise reduction that involves the treatment of the signal samples in order to make them fit a particular model. Smoothing consists of adjusting sample by sample the signal based on a regional polynomial fit. With smoothing, you do not have to design a filter that is robust to outliers, can easily adapt to varying sampling rate, and preserve the sharpness of peaks while eliminating high-frequency components. However, smoothing is more computationally intensive than linear filtering.

There are two types of polynomial smoothing methods for mass spectra that remove the false ion peaks that do not indicate compounds in the sample. These methods preserve the sharpness (high-frequency components) of the ion peaks by smoothing the curve using nonparametric and polynomial filtering methods [25, 26].

4.6.1 Lowess Filter Smoothing

Lowess filters smooth a mass spectrum by using a locally weighted linear regression method. The smoothing process is considered local because each smoothed value is determined by neighboring data points within a span. The process is weighted because a regression weight function is defined for the data points contained within the span. The weight sequence is given by the tricube function shown below [27, 28].

$$w_i = \left(1 - \left| \frac{x - x_i}{d_x} \right|^3 \right)^3 \quad 4.3$$

The m/z vector might not be uniformly spaced. Therefore, the sliding window (span) for smoothing is centered using the closest samples in terms of the m/z value and not in terms of the m/z vector indices.

For example, if the span is 10 samples, the method consists of performing a locally weighted regression smoothing algorithm by applying a full least-squares fit with the 10 closest samples to the point to be fixed. This step is repeated for every point in the signal. One of its strengths lays in its ability to effectively adapt to data with nonuniformly spaced values.

A linear fit (Lowess) or a quadratic fit (Loess) is usually employed, but a zero order may also be used, which is equivalent to a weighted local mean estimator. Samples are weighted in the fitting process, which allows emphasis of those samples that are closest to the point being fixed. Different weighting approaches have been proposed such as using a tricubic function, a Gaussian pulse, or a triangle shape.

This polynomial fitting approach allows an estimate of how much you need to correct at every point. By doing some statistics on this data, it is easy to detect potential outliers which you can simply remove from the signal. This allows reapplying the algorithm until no more outliers are detected, and recalling that the previous procedure in the algorithm does not require evenly spaced samples [29].

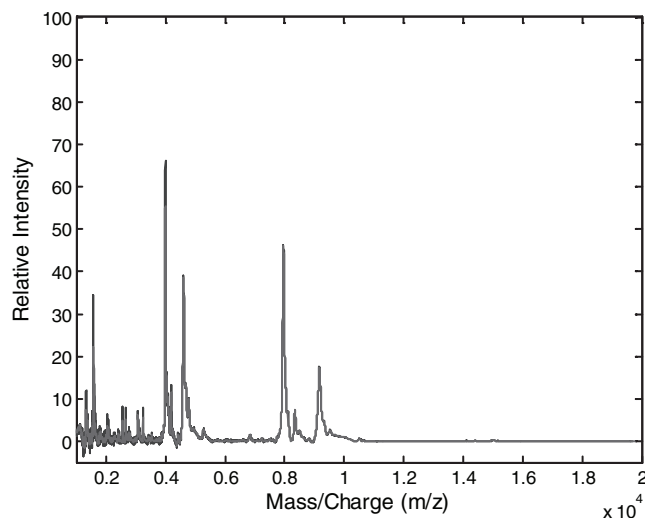


Figure 4.17 Smoothed spectrum using a least-squares polynomial filter.

4.6.2 Savitzky and Golay Filter Smoothing

Savitzky and Golay filters smooth a mass spectrum using a least-squares digital polynomial filter. The Savitzky and Golay method of smoothing is a generalization of the Lowess method. You derive the filter coefficients by performing an unweighted linear least squares fit using a polynomial of a given degree. It allows you to use higher order polynomials for the fitting. As a result, the algorithm preserves signal features such as the resolution between ion peaks and the height of the peaks. The original algorithm by Savitzky and Golay assumes a uniformly spaced mass/charge vector while the function `mssgolay` also allows one that is not uniformly spaced [30].

One of the most important parameters in polynomial filtering is the size of the window, (or spanning). It is indirectly associated with the cut-off frequency. However, there is not a practical relation between these two so you can usually adjust the window based on experimental experience. For example, in a low resolution mass spectrum signal, it is common to have the span set to 15-20 samples.

4.6.3 Example Demonstrating Noise Smoothing

Smooth the normalized spectra with a polynomial filter of second order. Most of the mass spectrometry preprocessing functions in the Bioinformatics Toolbox have an input parameter `Showplot` that creates a customized plot to help you follow and assess the quality of the preprocessing action. See Figure 4.17.

```
YS = mssgolay(MZ, YN2, 'SPAN', 35, 'ShowPlot', 3);
```

Zooming into a reduced region reveals the detail of the smoothing algorithm. See Figure 4.18.

```
axis([8000 9000 -1 8])
```

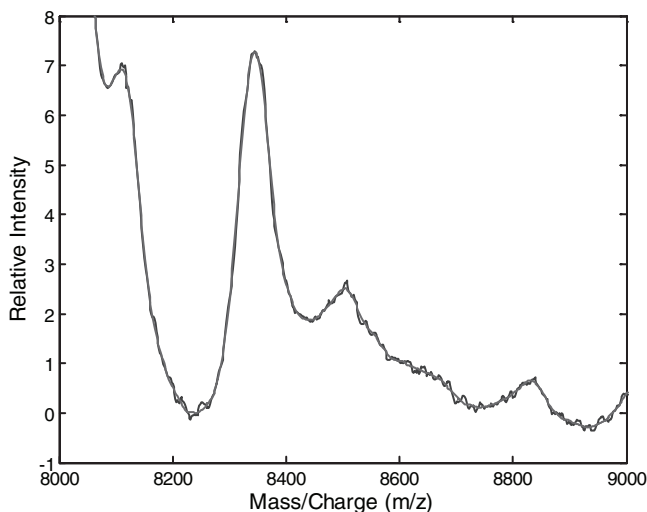


Figure 4.18 Detail showing noisy and smoothed spectrum.

4.7 Identifying Ion Peaks

After you have adjusted the baseline, corrected for calibration, normalized the intensities, and smoothed the spectra, you can identify peaks.

A simple approach to finding putative peaks is to look at the first derivative of the smoothed signal.

```
slopeSign = diff(YS(:,1)) > 0;
slopeSignChange = diff(slopeSign) < 0;
h = find(slopeSignChange) + 1;
```

Remove peaks in the low-mass region below 1500 Daltons and small ion intensity peaks with a height below 5.

```
h(MZ(h) < 1500) = [];
h(YS(h,1) < 5) = [];
```

Plot the spectrum with identified peaks.

```
plot(MZ,YS(:,1),'- ',MZ(h),YS(h,1),'ro');
```

See Figure 4.19 showing the ion peaks detected in a spectrum.

More elaborate peak detection methods use discrete wavelet transforms (DWT) for isolating the noise, and then finding the putative peaks. When using DWT special care needs to be taken to account for signal shifts and varying signal resolution [31].

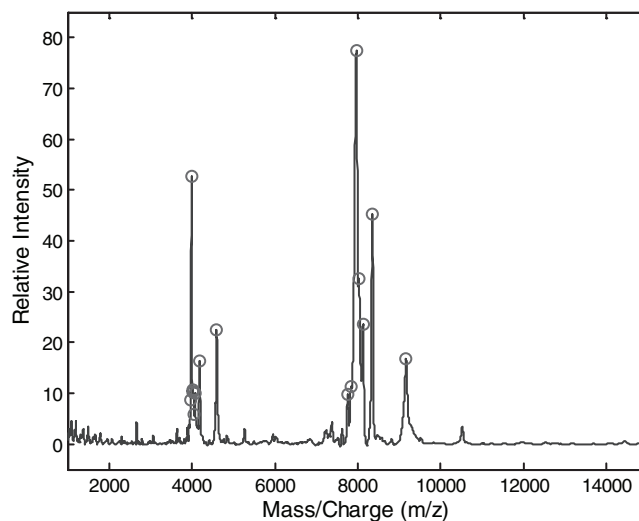


Figure 4.19 Spectrum with identified ion peaks.

References

- [1] Collins, F. S., et al., "New goals for the U.S. human genome project: 1998–2003," *Science*, Vol. 282, No. 5389, 1998, pp. 754–759.
- [2] Mullikin, J. C., and A. A. McMurray, "Sequencing the genome, fast," *Science*, Vol. 283, No. 5409, 1999, pp. 1867–1868.
- [3] Nelson, D. O., "Improving DNA sequencing accuracy and throughput," in *Genetic Mapping and DNA Sequencing*, New York: Springer, 1996.
- [4] Draghici, S., *Data Analysis Tools for DNA Microarrays*, New York: Chapman & Hall/CRC, 2003.
- [5] Aldred, S., M. M. Grant, and H. R. Griffiths, "The use of proteomics for the assessment of clinical samples in research," *Clin. Biochem.*, Vol. 37, No. 11, 2004, pp. 943–952.
- [6] Conrads, T. P., et al., "Cancer diagnosis using proteomic patterns," *Expert Rev. Mol. Diagn.*, Vol. 3, No. 4, 2003, pp. 411–420.
- [7] Zhang, Z., et al., "Three biomarkers identified from serum proteomic analysis for the detection of early stage ovarian cancer," *Cancer Res.*, Vol. 64, No. 16, 2004, pp. 5882–5890.
- [8] Fenn, J. B., et al., "Electrospray ionization for mass spectrometry of large biomolecules," *Science*, Vol. 246, No. 4926, 1989, pp. 64–71.
- [9] Tanaka, K., et al., "Protein and polymer analyses up to m/z 100 000 by laser ionization time-of flight mass spectrometry," *Rapid Commun. Mass. Spectrom.*, Vol. 2, No. 8, 1988, pp. 151–153.
- [10] Liebler, D. C., *Introduction to Proteomics: Tools for the New Biology*, Humana Press, 2001.
- [11] Gentleman, R., et al., *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, New York: Springer, 2005.
- [12] Baggerly, K. A., J. S. Morris, and K. R. Coombes, "Reproducibility of SELDI-TOF protein patterns in serum: comparing data sets from different experiments," *Bioinformatics*, Vol. 20, No. 5, 2004, pp. 777–785.
- [13] Sorace, J. M., and M. Zhan, "A data review and re-assessment of ovarian cancer serum proteomic profiling," *BMC Bioinformatics*, Vol. 4, 2003, pp. 24.
- [14] Petricoin, E. F., et al., "Use of proteomic patterns in serum to identify ovarian cancer," *Lancet*, Vol. 359, No. 9306, 2002, pp. 572–577.
- [15] Institute, N. C., *FDA-NCI Clinical Proteomics Program Databank*, <http://home.ccr.cancer.gov/ncifdaproteomics/>.
- [16] MathWorks, *Bioinformatics Toolbox Reference*, Natick, MA: MathWorks, 2005.
- [17] MathWorks. *Bioinformatics Toolbox Demonstration*, 2005 [cited; available from http://www.mathworks.com/products/demos/bioinfo/massspec_prepro/mspreprodemo.html].
- [18] Andrade, L., and E. Manolagos, "Signal background estimation and baseline correction algorithms for accurate DNA sequencing," *J. VLSI Signal Processing Systems*, Vol. 35, No. 3, 2003, pp. 229–243.
- [19] Jeffries, N., "Algorithms for alignment of mass spectrometry proteomic data," *Bioinformatics*, Vol. 21, No. 14, 2005, pp. 3066–3073.
- [20] Du, P., W. A. Kibbe, and S. M. Lin, "Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching," *Bioinformatics*, Vol. 22, No. 17, 2006, pp. 2059–2065.
- [21] Wagner, M., D. Nalk, and A. Pothen, "Protocols for disease classification from mass spectrometry data," *Proteomics*, Vol. 3, No. 9, 2003, pp. 1692–1698.
- [22] Satten, G. A., et al., "Standardization and denoising algorithms for mass spectra to classify whole-organism bacterial specimens," *Bioinformatics*, Vol. 20, No. 17, 2004, pp. 3128–3136.

- [23] Li, L., et al., "Application of the GA/KNN method to SELDI proteomics data," *Bioinformatics*, Vol. 20, No. 10, 2003, pp. 1638–1640.
- [24] Lilien, R. H., H. Farid, and B. R. Donald, "Probabilistic disease classification of expression-dependent proteomic data from mass spectrometry of human serum," *J. Comput. Biol.*, Vol. 10, No. 6, 2003, pp. 925–946.
- [25] Bowman, A. W., and A. Azzalini, *Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-plus Illustrations*, London: Oxford Univ. Press, 1997.
- [26] Orfanidis, S. J., *Introduction to Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1996.
- [27] MathWorks, *Curve Fitting Toolbox Reference*, 2005, Natick, MA: MathWorks.
- [28] Cleveland, W. S., and S. J. Devlin, "Locally-weighted regression: an approach to regression analysis by local fitting," *J. Amer. Statistical Assoc.*, Vol. 83, No. 403, 1988, pp. 596–610.
- [29] Cleveland, W. S., "Robust locally weighted regression and smoothing scatterplots," *J. Amer. Statistical Assoc.*, Vol. 74, 1979, pp. 829–836.
- [30] Savitzky, A., and M. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Anal. Chem.*, Vol. 36, 1964, pp. 1627–1639.
- [31] Coombes, K. R., et al., "Improved peak detection and quantification of mass spectrometry data acquired from surface-enhanced laser desorption and ionization by denoising spectra with the undecimated discrete wavelet transform," *Proteomics*, Vol. 5, No. 16, 2005, pp. 4107–4117.

PART III

Analysis: Control and Systems

Control and Systems Fundamentals

Fulvia Ferrazzi and Riccardo Bellazzi

5.1 Introduction

All cells contain DNA, RNA, proteins and other smaller molecules involved in signaling and energy transfer and thus function as biochemical factories of a broadly similar type. Major questions in biology are: how do all these components act together to respond to environmental signals and how does the vast variety of cell types develop? Traditional biology approaches usually study the function of a single gene or protein; however, the response of a cell to an environmental cue is highly complex and is the result of the interactions among many components. These concepts are at the basis of the research field known as “systems biology” [1].

The systems approach to biology benefits from the massive amount of data that is being generated by genomics and proteomics high-throughput technologies. This data provides a global view of gene and protein expression patterns of cells after certain environmental signals, thus offering a unique opportunity to characterize cellular responses to disease and stress, as well as to monitor developmental regulatory processes [2]. Acquiring the data is only the start, as the results later need to be managed and interpreted; a fundamental aspect of systems biology is the development of mathematical models that exploit experimental data in order to understand the complex relationships and interactions among the components of cellular systems.

An engineering approach to the study of cellular systems appears highly appropriate and promising [3–6]. From a systems theory viewpoint, a cell is a dynamical system, which can be completely characterized by a set of variables, called state variables; the state variables of a cellular system may be the whole set of gene or protein expression levels. The cell can also be described as an input/output (I/O) system, composed of simpler interconnected components and able to provide a response, or output, to external stimuli or controlled manipulations, called inputs. Inputs to a cellular system can be physical (e.g., temperature changes or a mechanical stress) or chemical (e.g., drugs, hormones, or nutrients), while measurable outputs can be signals to other cells or the activation of transcription factors. Subsystems within a cell can be identified as being involved in different processes, such as cell growth and maintenance, division, and death [7].

The cell is not only a dynamical system, but it is also a controlled environment. The state variables are typically interconnected in a way able to maintain the basic cell functionalities, even in the face of possible disturbances. The analysis of cellular control mechanisms may highlight cell robustness or fragility with respect to the different challenges it has to face. Moreover, it could allow the design of proper pharmacological interventions in order to obtain a desired effect. In this context, the use of concepts and techniques from the branch of systems theory known as control theory may be very useful. In particular, genome-wide data enables the modeling of the system at a level of detail that may be sufficient to unravel some of its control mechanisms and to consequently lead to plan specific interventions.

This chapter aims at giving the reader basic tools to interpret current research efforts in systems biology from an engineering perspective. In the beginning of the chapter the fundamental concepts of control and systems theory are reviewed (Section 5.2) and the application of these concepts to the analysis of biological systems discussed (Section 5.3). Then, the challenging task of reverse engineering cellular networks, that is, trying to infer the relationships between cellular variables from genome-wide data, is introduced (Section 5.4). The rest of the chapter is devoted to methods to infer gene networks from gene expression time series (Section 5.5), focusing on Boolean networks, one of the simplest models to describe gene interactions (Section 5.5.1), and on dynamic Bayesian networks, a more advanced technique (Section 5.5.2).

5.2 Review of Fundamental Concepts in Control and Systems Theory

A control system may be defined as a set of interconnected components that act together to maintain a desired behavior despite the action of external disturbances. As an example, the task of controlling the temperature of a room in winter, using a fan-heater that heats and circulates air across a room, is considered [8]. In this case the desired behavior is the maintenance of the temperature within a specific range.

There are two basic ways in which a control system can operate. In *open-loop* mode the system is controlled without using any information about the output. In the example under consideration, this would mean trying to control the room temperature by setting a specific level of the fan-heater. In this way, however, if the external temperature rises, the room is going to become warmer, because the amount of heat introduced by the heater is now greater than that of the heat dissipated from the room. Similarly, when the external temperature falls, the room temperature is going to decrease.

For this reason, biological systems and engineering plants are typically controlled in *closed-loop* mode. The general idea in this case is that “the output of the system is fed back and used to adjust the system input” [8]. In the example this could be achieved by measuring the room temperature, comparing it with the desired temperature and adjusting the heater setting in proportion to their difference. Other strategies that make use of the measurement of the room temperature can be used; the common feature of all of them is *feedback*, which the mathematician-engineer N. Wiener defined as “a method of controlling a system by reinserting into it the results of its past performance.” Feedback control is highly common in na-

ture: homeothermic animals, for example, employ a temperature controlling strategy analogous to the one described above.

The open-loop and the closed-loop modes are schematically represented in Figure 5.1 and Figure 5.2. In both cases the general objective of the control system is that the time dynamics of a certain *controlled variable* in a *system (plant)* coincides with that of a preset *reference variable*, despite the action of some *disturbances*. In order to achieve this goal it is necessary to act on a *control variable*, and this is done through the *controller*. In the example of the heating room, the heater is the controller and the room is the plant. In the open-loop system (Figure 5.1), the controller has no information about the value of the output (controlled variable) of the system (plant) and has therefore no means of compensating for the action of the disturbances that act on the system, affecting its output. In contrast, in the closed-loop configuration (Figure 5.2), a *feedback sensor* measures the system output and compares the resulting measurement (*feedback signal*) with the desired reference variable. The deviation (*error*) is used to calculate the *control variable*. The example shown in Figure 5.2 is that of negative feedback: here the feedback signal is subtracted from the input reference variable. If the resulting error is positive, the controller acts on the system in order to increase the output; if instead the error is negative, the controller acts to decrease the output. Negative feedback is not the only available feedback configuration: *positive feedback* is also possible, in which the feedback signal is added to the reference input; in this configuration, the higher the output compared to the reference input, the higher the error, which in turn increases the output even further. Even if it may seem dangerous, positive feedback is employed in various physiological processes, such as the propagation of action potentials in neuronal dynamics [8].

Control theory is highly interwoven with the theory of dynamical systems. In order both to analyze the properties of a given control system and to design a specific control system according to certain requirements, it is necessary to have a mathematical model of the system under analysis. Cellular systems are no exception; only an effective mathematical description of a cellular system could allow the study of its properties and the design of appropriate interventions that could eventually be used to force the transition from a diseased cellular state to a healthy one.

A “dynamical system” is described with a mathematical model of a physical/biological entity that interacts with the environment through two vectors of time-dependent variables. The former, called *input variables*, represent the actions performed on the system by external agents that influence its behavior; the latter, called *output variables*, represent the observed reaction of the system to its inputs.

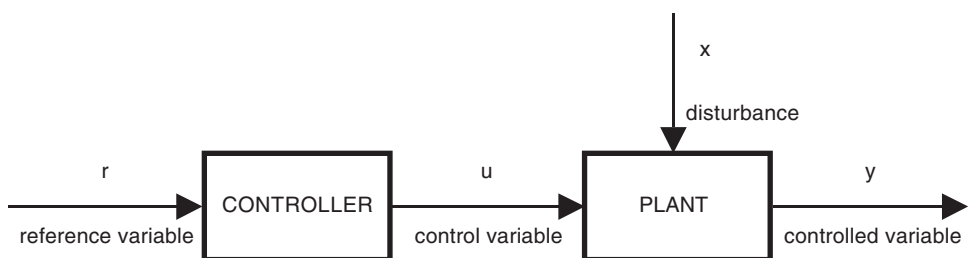


Figure 5.1 Control system in open-loop mode.

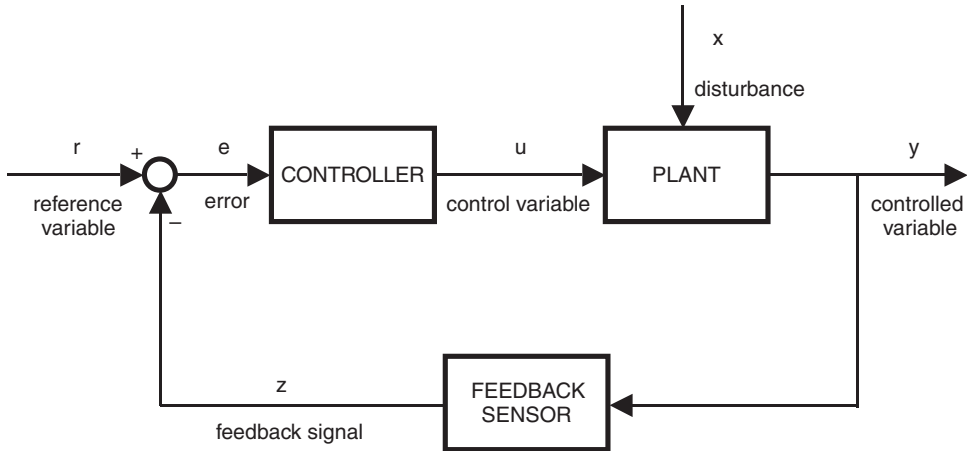


Figure 5.2 Control system in closed-loop mode.

In a dynamical system the knowledge of the value of the input variables is usually not sufficient to determine the value of the output variables at the same time. Suppose, for example, one is interested in filling a bathtub with water; the output one needs to control is the level of water in the bathtub. It then appears clear that knowing only the quantity of water that flows through the tap is not enough to predict when the tub is going to be full: one also needs to know the initial level of water. It is therefore necessary to introduce also the so-called *state variables*, defined as the minimal set of variables able to characterize the system, summarizing its “past history,” that is, capturing its evolution up to time t . In order to predict the future output of a deterministic dynamical system with given functional relationship between the inputs and the outputs, it is necessary and sufficient to know the current values of the state and input variables.

Supposing the *dimension* (number of state variables) of a system is n , a *state-space* representation of this system is the set of equations describing its behavior in the n -dimensional space (R^n) of the state variables. Indicating with $u \in R^m$, $x \in R^n$, and $y \in R^p$ the vectors of input, state, and output variables, the state-space representation of a continuous-time dynamical system is given by the following equations:

$$\dot{x}(t) = f[x(t), u(t), t] \quad 5.1$$

$$y(t) = g[x(t), u(t), t] \quad 5.2$$

Equation (5.1) (*state equation*) is a differential equation that defines the evolution of the state $x(t)$, once the initial state x_0 and the input function $u(t)$ are known. Equation (5.2) (*output transformation*) is an algebraic equation that allows determining the output as a function of the input and the state.

A linear system is a system for which the functions f and g are linear so that $\dot{x}(t)$ and $y(t)$ can be expressed as a linear combination of $x(t)$ and $u(t)$. Roughly speaking it is possible to say that a linear system is a system where, if the input is changed by a factor δ , the change in the output is proportional to δ . Considering, for example, an ideal spring/mass system (Figure 5.3), if the mass is pulled twice as far down, then the corresponding oscillation will be twice as large. A number of well-established techniques are available for the analysis of linear systems, such as tech-

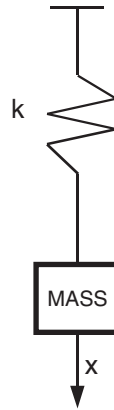


Figure 5.3 Ideal spring/mass system. The system responds linearly to variations in the input.

niques based on spectral analysis. Yet, the vast majority of both natural and artificial systems are nonlinear, and their analysis requires more complicated tools [9].

Both linear and nonlinear dynamical systems share some fundamental properties. An important one is that of *stability*. Given a certain initial state x_0 at $t = t_0$ and an input function $u(t)$ for $t \geq t_0$, the *state response* of the system is defined as the sequence of values taken by its state during time, while the *state trajectory* is the projection of the state response in the state space R^n . The stability of a system is a property of its state responses corresponding to different initial states: a state response is said to be stable if it is unaffected by small perturbations. Stability is usually categorized as *internal stability*, related to the perturbations of initial conditions; *external stability*, related to perturbations of the system inputs; and *structural stability*, related to changes in the system parameters. In a linear system all state responses are either stable or unstable, so that the system itself can be defined as globally stable or unstable.

Dynamical physical systems are usually dissipative; once an input, which was applied for a finite time interval, returns to zero (so that the system is said to be autonomous), the output, after a transient period, will present a typical behavior. The region of the state space corresponding to such typical behavior is called *attractor*. Attractors can therefore be found from the solutions of (5.1) when $u(t) = 0$ and $t \rightarrow \infty$. Very common kinds of attractors are the equilibrium points or steady states; these are, by definition, constant state responses obtained by solving (5.1) with $\dot{x}(t) = 0$. Stability of an equilibrium is a particular case of the stability of a state response: an equilibrium point $\bar{x}$ may be stable, when small perturbations give rise to trajectories that remain close to $\bar{x}$ and tend to $\bar{x}$ when $t \rightarrow \infty$; unstable, when, after small perturbations, the trajectories diverge from $\bar{x}$; and simply stable in all other cases.

The concept of stability is strictly linked to the concept of *robustness* of a system. Robustness is “the degree to which a system or component can function cor-

1. *IEEE Standard Computer Dictionary*, The Institute of Electrical and Electronics Engineers, New York, USA, 1990.

rectly in presence of invalid inputs or stressful environmental conditions.”¹ It refers to the existence of a standard working condition of the system and to its ability to maintain this condition despite perturbations. Feedback, in particular negative feedback, is the most common means used to achieve robustness and stability, as it is able to compensate for the uncertainty present in the system and to maintain the system output within a predefined range of acceptable values. The issue of stability and robustness in biological systems has received increasing attention from the engineering community and is thought to have been “a key starting point in developing a conceptual and theoretical bridge to biology” [4].

5.2.1 Discrete-Time Dynamical Systems

During biological and medical experiments, the measurements are typically collected over a finite set of time points. This situation is modeled by assuming that at a given time point t_k the output function y is sampled, thus obtaining a measurement z_k . If the measurements are affected by experimental error, a stochastic variable $v(t_k)$ may be introduced into the model, so that the analyzed dynamical system is still described by (5.1) and (5.2), while the available measurements are described as

$$z_k = y(t_k) + v(t_k) \quad 5.3$$

where t_k belongs to the vector of measurement times $[t_1, t_2, \dots, t_N]$ and N is the number of measurements.

Another suitable modeling solution, widely used in bioinformatics and systems biology, is to exploit discrete-time dynamical systems. Unlike continuous-time systems seen so far, in which the state variables were defined for any real value of the time variable t , in discrete-time systems the states are defined only for integer time points $t = (0, 1, 2, \dots)$. In this case, the state equation and output transformation are

$$x(t+1) = f[x(t), u(t), t] \quad 5.4$$

$$y(t) = g[x(t), u(t), t] \quad 5.5$$

The equations above, as well as (5.1) and (5.2), describe a deterministic dynamical system. However, since deterministic models are often insufficient to capture the complexity of biological environments, different stochastic counterparts of (5.4) and (5.5) have been proposed. In the simplest case seen above, that is, if the measurements are noisy, we can again introduce a suitable stochastic variable $v(t)$ into the model, so that (5.5) becomes

$$y(t) = g[x(t), u(t), t] + v(t) \quad 5.6$$

It is also possible that the process governing the dynamics of the system is not completely known or affected by unmodeled effects. In this case, an effective modeling strategy is to add a noise component also to the state equation, giving rise to a stochastic dynamical system:

$$x(t+1) = f[x(t), u(t), t] + w(t) \quad 5.7$$

$$y(t) = g[x(t), u(t), t] + v(t) \quad 5.8$$

The linear version of this model, known as the *Kalman filter*, has been widely used in engineering applications. The general class of stochastic models represented by (5.7) and (5.8) also encompasses hidden Markov models, very popular in bioinformatics applications [10]. Such models may be applied also when the number of possible values of the state variables are finite; in this case the time relationships are often represented by a transition probability matrix between state values. The dynamic Bayesian network formalism presented in Section 5.5.2 is a graphical representation of the stochastic dynamical system just introduced.

5.3 Control Theory in Systems Biology

Systems biology can highly benefit from the use of tools developed in the area of control theory. A cell is both a dynamical system and a controlled environment. Moreover, the paradigm of an input/output system composed of simpler interconnected components is very natural in the context of cellular systems. An important direction of current systems biology research is aimed at understanding cells' behavior, analyzing their subsystems (or modules) and how they act in concert to carry on the specific functions necessary for cell life [3, 11, 12]. Hartwell et al. affirm that it is precisely the notion of function or purpose that differentiates biology from the other natural sciences, at the same time bringing it nearer to synthetic sciences, such as computer science or engineering, in which the concept of function appears naturally [3].

Some design principles of biological systems coincide with the ones often employed in engineered control systems. The most striking example is feedback. Examples of positive feedback are the loops that drive both the entrance and the exit of cells to and from mitosis [13], while a well-studied example of negative feedback is given by the bacterial chemotaxis signaling pathway, in which a sensory system detects subtle variations in an input signal [14, 15]. Other examples of feedback control can be found in genetic networks. These networks “withstand considerable variations and random perturbations of biochemical parameters [that] occur as transient changes in, for example, transcription, translation, and RNA and protein degradation” [16]. In the context of gene regulation, one talks about feedback whenever a protein modifies, either directly or indirectly, its own production rate. Various research work has focused on examining the stability properties of gene networks dominated by positive versus negative feedback; a central result was that “genes regulated by negative feedback should be more stable than unregulated genes or those regulated by positive feedback” [17].

In addition to feedback regulatory structures, stability and robustness are other two key features that characterize both biological and engineered systems. A very interesting review on this subject is provided by Stelling et al. [11]. The authors explain how the “robust yet fragile” nature of biological systems is strictly linked to their complexity. A significant example is given by cancer: here fragility at the cellular level (apoptosis of cells carrying dangerous mutations) allows the organism to be robust; conversely, cellular robustness (uncontrolled growth of neoplastic cells) can be very risky for the organism. The authors then consider the set of mechanisms

that confer robustness to both biological and engineered systems. One of these is feedback: a balance of negative and positive feedback allows a system to be both sensitive and stable.

An interesting example of how to analyze a biological system carried out from an engineering viewpoint is given by El-Samad et al. [6, 18] and discussed in [19]. The heat shock response in *E. coli* was studied: this response is activated when a cell is exposed to very high temperatures, an extremely dangerous situation as heat is able to denature proteins. Heat shock induces the production of “heat shock proteins” that help to refold denatured proteins and to degrade those that can be harmful for the cell. El-Samad et al. first constructed a full mathematical model to describe the dynamics of each signal over time. They then decomposed this model, identifying the functional modules typical of traditional engineering control architectures: the plant (here, the refolding of denatured proteins), the controller (the level of an RNA polymerase cofactor, whose activity increases after heat shock), and open loop and closed loop mechanisms. In this way they developed a reduced mathematical model able to describe the dynamics of each module. The analysis of this smaller model led El-Samad et al. to some simulation experiments on the larger model, aiming to compare the performance of the closed-loop and the open-loop configurations. The authors proved the increased robustness to parameter variability and the property of noise attenuation given by the feedback configuration. This example shows how the application of control principles to the analysis of a biological system is able to produce an intuitive representation of the system that can offer interesting insights about its underlying architecture. This could allow one to predict the response of the system under unknown conditions and to reengineer the system in order to achieve a desired behavior [19].

Interestingly, not only can systems biology significantly benefit from the use of control theory techniques, but also the converse is true, as totally new theoretical control questions arise from the study of biological systems. E. D. Sontag addresses these issues in his papers: the author’s main point is that often problems in systems biology resemble standard problems in control theory but, if examined more carefully, they actually show some fundamental differences that are worth exploring [5, 7]. For example, a significant challenge in systems biology is encountered when analyzing signaling networks. Traditional techniques to model these complex systems would require the use of biological knowledge to design large-scale simulation models. However, estimating the model parameters *in vivo* is very hard also in principle, as the concentrations of enzymes and others chemicals vary a lot from one cell to the other. These significant experimental limitations raise the need for more effective theoretical tools. The paradigm of decomposition and reconnection typical of control engineering can be exploited; the signaling system is decomposed into several subsystems and, from the study of these, the behavior of the entire system is reconstructed. A new class of subsystems particularly suitable for the analysis of enzymatic cascades and feedback loops was identified and called monotone I/O systems [20].

Other novel theoretical studies in control engineering are stimulated by the analysis of robustness in cellular systems. While few engineered systems work well under large variations in their parameters, living cells perform satisfactorily even in the presence of significant variations in the concentration of their chemicals. Evo-

lution must have acted to select for extremely robust structures, and their study is highly interesting, as it can suggest novel designs for engineering applications.

5.4 Reverse Engineering Cellular Networks

Another very active area of research in systems biology, that also raises new control-theoretic issues, is that of reverse engineering cellular networks: from the measurements of certain variables, such as protein concentrations or amounts of transcribed RNA, one tries to infer the internal structure of the cellular system. While this topic seems a perfect target for traditional system identification techniques, a number of issues distinguish it from more standard formulations of systems identification [5, 7]. First of all, in most cases it is either very expensive, or even unfeasible, to apply arbitrary test signals to the system. This implies that it is not possible to gather enough information to characterize the behavior of the highly nonlinear biological systems. Another important problem is related to the fact that often only steady-state measurements are available, that is, measurements of the system in a stable condition. In traditional genetic experiments or pharmacological interventions, when a perturbation to a gene or a signaling component is applied, it rapidly propagates through the network so that only a “global response” can be observed, after the system has already reached a new steady-state. For example, cells respond to growth factors stimulation with transient changes, like phosphorylation, that last only a couple of minutes. This time frame makes it unfeasible to gather enough intermediate data points to model the transitions that take place in the cell. Studies that address this issue can be found in the literature [21, 22].

Various recent research efforts are aimed at trying to reverse engineer cellular systems by analyzing high-throughput data. This data can provide more information about the internal state of a cell than what is possible using standard biological techniques, thus improving the chances to unravel cellular control mechanisms.

The development of genome-wide technologies was strictly linked to the Human Genome Project [2]. DNA sequencing and genotyping techniques (i.e., the analysis of the genetic polymorphisms in an individual DNA sample) enable the development of tools to identify the genetic variations associated with certain observed phenotypes. For example, single-nucleotide polymorphisms (SNPs), the variations of a single base among the individuals in a population, are a promising tool to discover the genetic bases of common diseases and arrays able to genotype thousands of SNPs at a time are now available [23]. Another useful technology to analyze the genomic structure of a cell is given by comparative genomic hybridization (CGH) microarrays, which provide genome-wide identification of chromosomal abnormalities, such as deletions and amplifications, frequently encountered in tumors [24].

The functional counterparts of these technologies are DNA microarrays, which enable investigators to measure the expression levels of thousands of genes at a time [25–28]. There are two main types of arrays: *cDNA microarrays*, introduced into common use at Stanford University and first described by Schena et al. in 1995 [29], and *oligonucleotide microarrays*, developed by Affymetrix of Santa Clara under the trademark GeneChip® [30]. Both types of arrays use the abundance of the mRNA

produced during the transcription phase as a quantitative measurement of expression level; this mRNA is later translated into proteins and therefore its measurement gives information about the activity of a cell. The initial step in the use of microarrays consists of extracting the mRNA contained in the cells of a biological tissue of interest and *reverse transcribing* this mRNA into a complementary DNA copy (cDNA), introducing a fluorescent label; this constitutes the so-called *target*. A microarray is a chip containing an ordered sequence of spots, with a diameter of less than 200 μm , in which single-stranded DNA sequences corresponding to a given gene are placed. These DNA portions, called *probes*, can be either cDNA sequences (cDNA microarrays) or short specific segments, known as synthetic oligonucleotides (oligonucleotide microarrays). The principle exploited by microarrays is that of *hybridization*, the coupling of complementary bases; the target binds to the complementary probes contained in the spots on the microarray. After removing the nonhybridized target, the microarray is put under a laser light and, by means of a digital scanner, the brightness of each fluorescent spot is measured. Studies have demonstrated that this brightness is correlated with the absolute amount of mRNA in the original sample and, by extension, to the expression level of the gene associated with this mRNA [29]. An advantage of cDNA microarrays is that it is possible to hybridize on the same array cDNA samples from two different tissues, labeling them with different dyes (see Figure 5.4) [31]. On the other hand, oligoarrays significantly mitigate cross-hybridization effects (the hybridization with se-

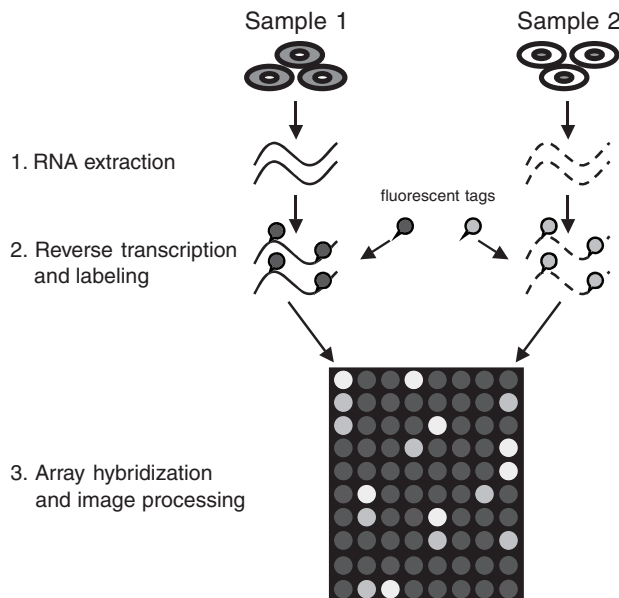


Figure 5.4 Schematic representation of the use of cDNA microarrays. Cellular mRNA is extracted from two samples, reverse transcribed into cDNA, and labeled with fluorescent tags of different colors. This target is then hybridized onto the microarray, then the brightness of each fluorescent spot is measured. These measurements can be represented in a colored image that shows spots in which the fluorescent intensity is higher in one sample than the other. (For a full-color figure, please refer to the CD.)

quences partially but not entirely complementary to the target), require a smaller amount of total RNA to prepare the target, and are able to also detect targets with very low abundance [32].

Microarrays offer the opportunity to study interactions among genes and to characterize disease states on the basis of their expression profiles. The measurement of expression levels at different development stages, or in different tissues, clinical conditions, or organisms is very useful to understand gene function, comprehend biological processes, and analyze the effects of pharmacological treatments. This technology has allowed, for example, for discriminating different tumor classes on the basis of their gene expression profiles more effectively than through analyses based only on tumor morphology [33, 34].

The field of proteomics takes genome-wide studies to the protein level; using mass spectrometry technology, investigators can now measure in parallel the entire protein content in a cell, tissue, or organism of interest [35]. These methods, however, are not yet as widespread as methods to profile gene expression.

It is necessary to keep in mind that, even if the contribution of each high-throughput technology to the advancement in biomedical research appears extremely significant, a deeper understanding of cellular processes seems possible only through the integration of data coming from different sources. Each technology is in fact able to offer only a partial view of the highly nonlinear dynamical processes that take place in a cell. In the literature significant examples of analyses that combine various types of genome-wide data can be found: for example, Hartemink et al. used both gene expression data and genome-wide location data (measurements of protein-DNA interactions) for gene network inference [36, 37]; Segal et al. studied transcriptional modules exploiting gene expression and promoter sequence data [38] and proposed an approach to identify molecular pathways from gene expression and protein interaction data [39]; and Nariai et al. presented a method for simultaneously reconstructing gene and protein networks using gene expression, protein-protein interaction, and other genome-wide data [40].

5.5 Gene Networks

Gene networks are being increasingly used as models to represent phenomena at the level of gene expression, that is, how the expression level of a gene affects the expression level of the others [41]. These networks offer a large-scale view of the state of the cell at the mRNA level, describing a large number of interactions in a concise way. Here the focus is on reverse engineering gene regulatory networks starting from gene expression time series, that is, expression measurements taken over time. Only temporal profiles give information about the dynamics of a cell's regulatory mechanisms and therefore their study is considered very promising for the discovery of functional relationships among genes [42].

Gene networks are often referred to as “gene regulatory networks,” even if the choice of the term “regulatory” is not the most appropriate: networks inferred from microarray data provide in fact only phenomenological descriptions, such as “every time that gene A is overexpressed, gene B is underexpressed.” The relationship

between the expression levels of the two genes can be the result of various mechanisms. If gene A codes for a transcription factor of gene B, then the relationship observed at the expression level is the result of an actual regulatory action of one gene on the other. Very often, though, the inferred relationship can be due to an indirect regulation of one gene on the other (gene A regulates gene C, which in turn regulates gene B) or can even be the effect of a common, unobserved cause (gene D regulates genes A and B).

Even if reverse engineering methods applied to DNA microarray data do not allow the inference of the whole set of actual regulatory processes, they constitute an important first step towards this goal. They indeed help identify sets of genes linked by potential cause-effect relationships, thus suggesting a number of novel biological hypotheses that can later be validated with ad hoc analyses. Moreover, gene networks provide a model for the dynamics of gene expression; assuming that the state variables of the system are given only by the set of measured gene expression values, this model can be employed to make predictions about changes in gene expression under certain experimental conditions.

In recent years, various methods for the inference of gene regulatory networks from DNA microarray data have been proposed. The majority of these methods aim at reconstructing both an *interaction network*, which encodes the links among genes, and a *dynamic model* of the interactions, able to describe the dynamics of the system [43]. The interaction network is usually represented by a graph. A graph is defined as a tuple $\langle V, E \rangle$, where V is a set of vertices and E a set of edges, while an edge is a tuple $\langle i, j \rangle$ of vertices that expresses the presence of a connection between two nodes i and j . If the graph is oriented, the arcs are directed and the tuple $\langle i, j \rangle$ indicates that the arc starts from node i and terminates into node j ; in this case one says that i is a parent for j .

A graph can be constructed following different approaches; a simple one is employed for relevance networks [44]. These networks are an intermediate solution between regulatory networks and clustering as they do not provide a model for the dynamics of the system. Clustering aims at grouping genes with a similar expression profile; this can be very useful for the inference of shared regulatory inputs and functional pathways; however, clustering does not say how the different gene groups interact with each other and who is regulating whom [45]. The basis for the construction of relevance networks is, like most clustering algorithms, the calculation of a correlation measure between gene profiles. Supposing the expression measurements for N genes in different experimental conditions or consecutive time points are available, the pairwise Pearson correlation for each couple of genes is calculated. By applying a properly chosen threshold to the absolute value of the correlation index, it is then possible to infer a nonoriented graph in which highly correlated genes are linked.

More advanced and widely used gene network models include Boolean networks, Bayesian networks, and methods based on differential equations. An extensive review of these methods is beyond the scope of this chapter and is available in the literature [43, 45]. Therefore it has been decided to use the Boolean network algorithm presented in [46] as a case study to exemplify the basic features shared by gene network learning algorithms, after which a Bayesian network approach for the study of dynamic data is presented.

Boolean network model, it is necessary to observe all the possible 2^N state transitions. For a realistic number of genes, it is very unlikely that all these configurations can be observed. If instead $K < N$, the data requirement significantly decreases and scales as $2^K(K + \log N)$. Additional constraints on the type of Boolean functions used can reduce the number of needed data points even further.

One of the most successful methods for the induction of Boolean networks from gene expression data is represented by the algorithm REVEAL (REVerse Engineering ALgorithm) by Liang et al. [46]. For each gene x , REVEAL considers all possible combinations of K inputs until it is able to find a set that unequivocally describes the output relative to gene x . In the search for the input set, the algorithm exploits the information theory concepts of entropy and mutual information; the logical function is then determined by comparing the state transitions with the Boolean function definitions. In the following section, the main features of the algorithm are presented.

5.5.1.1 Entropy and Mutual Information

Shannon *entropy* is a quantitative information measure. In this context “information” is used as a technical term and its meaning can be considered equal to “uncertainty.” The Shannon entropy H is defined in terms of the probability p_i of observing a particular symbol or event, within a given sequence:

$$H = -\sum_i p_i \cdot \log_2 p_i \quad 5.9$$

The entropy relative to the sequence of values of a binary variable x is therefore calculated as

$$H(x) = -p(0) \log_2 p(0) - p(1) \log_2 p(1) \quad 5.10$$

where $p(0)$ and $p(1)$ refer respectively to the probabilities of x being equal to 0 and 1. These probabilities are calculated as the frequencies of occurrences of the two binary states in the observed sequence.

Entropy H gives a measure of the uniformity with which x is distributed in the two different states: H is maximum when the states are equiprobable and decreases as the distribution in the different states becomes biased, as shown in Figure 5.6. $H = 0$ if all the occurrences are equal; this situation corresponds to a “no uncertainty” state, that is, “no information.”

In order to evaluate the relationship between two different variables, it is necessary to use a score that measures the information contained in the sequence of values of a variable with respect to the sequence of the other. Given two variables x (index i) and y (index j), their *joint entropy* is defined as

$$H(x, y) = -\sum_i \sum_j p_{i,j} \log_2 p_{i,j} \quad 5.11$$

$H(x, y)$ is therefore calculated from the frequencies of co-occurrences in the sequences.

The *conditional entropy* is then defined as

$$H(x|y) = H(x, y) - H(y) \quad 5.12$$

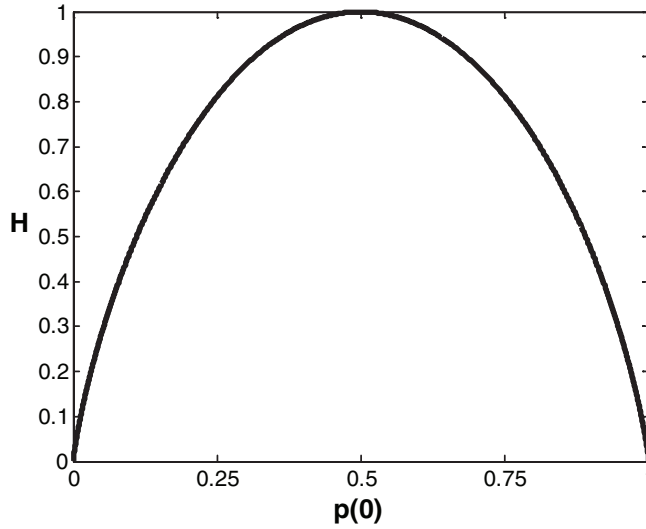


Figure 5.6 Shannon entropy for a binary variable, whose states are conventionally denoted as 0 and 1. The plot shows the entropy H as a function of the probability $p(0)$ of the variable being equal to 0.

and represents the information contained in x and not shared by y (that is, the remaining uncertainty in x , given knowledge of y). $H(y|x)$ is calculated in an analogous way.

Given these definitions, the *mutual information* $M(x, y)$ is the shared information between x and y . This can be calculated as the remaining information in x if the information in x that is not shared by y is removed (and similarly for y):

$$\begin{aligned} M(x, y) &= H(x) - H(x|y) \\ &= H(y) - H(y|x) \\ &= H(x) + H(y) - H(x, y) \end{aligned} \tag{5.13}$$

Figure 5.7 represents the above-defined scores with Venn diagrams.

If $M(x, y) = H(x)$, it means that y unequivocally determines x . This corresponds to the situation in which a certain value of x always corresponds to the same value of y . In an analogous way, it is possible to consider the entropy of x with respect to that of other two variables, y and z . If $M[x, (y, z)] = H(x)$, x is unequivocally determined by y and z (this is called interaction of order $K = 2$), and so on for $K = 3$, etc.

5.5.1.2 The Algorithm REVEAL

In gene expression time series, measurements of the gene expression levels at two consecutive time points correspond to an observed transition between two states of the network. It is important to remember that, in order to apply the algorithm REVEAL to gene expression data, the continuous expression values need first to be transformed into binary 0/1 values. Therefore, if the number of genes is, for example, $N = 3$, there are $2^3 = 8$ possible different state transitions, because each of the 8 possible states (expression values of the 3 genes) at time t unequivocally determines the state at time $t + 1$. An example of a complete state transition table is given

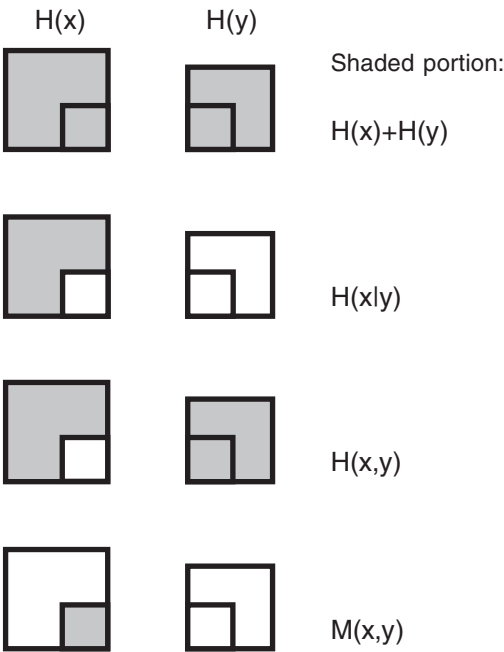


Figure 5.7 Venn diagram representation of entropy, conditional entropy, joint entropy, and mutual information. Image adjusted from S. Liang et al., *Pac. Symp. Biocomput.*, 1998.

in Table 5.1. This table is relative to a network with three genes A, B, and C, whose wiring diagram is represented in Figure 5.5.

For each gene x , the algorithm starts by looking for all the possible interactions of order $K = 1$. If no single gene that unequivocally determines x is found, then the algorithm searches in the space $K = 2$. Finally, if this search doesn't succeed either, the space $K = 3$ is explored. The search continues until an input set able to explain the examined output is found. Once the input set is chosen, the logical rule that allows calculating the output from the input is determined from the observed transitions. Of course, as K increases, the search becomes more computationally intensive.

Referring to the state transitions in Table 5.1 and considering, for example, gene A, the algorithm first checks whether $M[A(t + 1), X(t)] = H[A(t + 1)]$, where X is any of the three genes. As not all the state transitions for A can be explained

Table 5.1 Example of a complete state transition table for a network with three genes: A, B, and C. The wiring diagram of this network is represented in Figure 5.5.

Input			Output		
$A(t)$	$B(t)$	$C(t)$	$A(t+1)$	$B(t+1)$	$C(t+1)$
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	1	0

using only one input gene, the algorithm goes on to consider input sets composed of two genes. In this case, as $M\{A(t+1), [B(t), C(t)]\} = H[A(t+1)]$, the pair $[B(t), C(t)]$ is able to unequivocally explain $A(t+1)$.

The process exploited by REVEAL is unequivocal and exact if all the possible state transitions are observed. In real gene expression time series this is very unlikely, as these time series normally have few samples and therefore only a limited number of transitions is available. However, Liang et al. showed that, for $N = 50$ and $K = 3$, the analysis of incomplete state transition tables (100 state transition pairs out of the possible $2^{50} \cong 10^{15}$) is also able to reliably reproduce the original wiring diagram and the corresponding rules. The higher K is, the higher the number of state transition pairs needed to correctly infer the true network. Therefore, when analyzing gene expression time series, a low K should necessarily be used.

5.5.2 Dynamic Bayesian Networks

Boolean networks, as well as their dynamics, are completely deterministic. However, an intrinsic stochasticity exists in biological systems, due to random fluctuations in the values of certain variables. In order to overcome the limitations of Boolean networks, probabilistic Boolean networks extend the Boolean network concept to a probabilistic setting [48]. An extremely powerful alternative is given by the formalism of Bayesian networks (BNs). BNs have been widely employed in various fields, such as artificial intelligence and statistics, for the representation and the use of probabilistic knowledge and are becoming increasingly popular for the analysis of different types of genomic and proteomic data [2]. They offer a number of significant advantages over other methods: not only are they able to model stochasticity, but they can also incorporate prior knowledge and handle hidden variables and missing data in a principled way. BNs have been applied to the analysis of gene expression data [51–54], protein-protein interactions [55], and genotype data [56, 57]. The formalism of BNs is extensively presented elsewhere in this book, together with a discussion of some applications. In this chapter the focus will be on dynamic Bayesian networks (DBNs), a special class of BNs that models the stochastic evolution of a group of random variables over time and is thus especially suitable to study dynamic gene expression data, that is, time series of expression measurements. A traditional BN is able to offer only a static view of the system under analysis, useful if one is interested in modeling its steady state. DBNs can instead model how genes regulate each other over time. Moreover, as shown later, the use of DBNs allows one to overcome the inability of BNs to represent feedback loops, a key regulatory mechanism in biological systems.

DBNs are particularly suitable to model dynamical systems under uncertainty, since they can be used to represent the discrete-time stochastic models described by (5.7) and (5.8). As an example, Figure 5.8 shows the DBN representation of a Kalman filter with three state variables x , whose time evolution represents the dynamics of the system, and one output variable y . Variables x are usually called *hidden*, as they are accessible only indirectly through the observation of y .

Murphy and Mian in 1999 were the first to propose the use of DBNs for modeling time series of gene expression data: they reviewed different learning techniques but did not apply them to a real dataset [58]. The increasing availability of

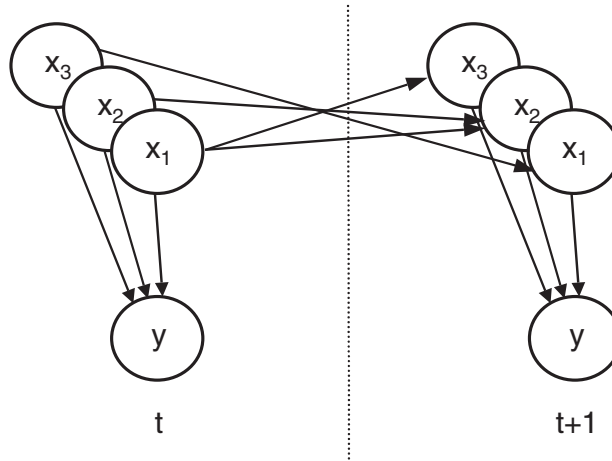


Figure 5.8 DBN representation of a Kalman filter.

microarray data has been accompanied by an increasing number of works in which DBNs were employed to analyze real gene expression datasets (see, for example [37, 59–63]). Moreover, as the evaluation of the inference results on a real dataset is controversial, detailed simulation studies have been performed in order to assess the performance of DBNs, as well as to test some advancements in the inference algorithms [64–66]. In the following section, a DBN approach based on linear Gaussian models is described. This approach is very useful for a first-level analysis of high-throughput dynamic data [67].

5.5.2.1 Linear Gaussian Networks

A DBN is a directed acyclic graph that encodes a joint probability distribution over a set of random variables: the nodes in the graph represent these stochastic variables and directed arcs represent the dependencies among them, which are quantified by conditional probability distributions. As in the case of BNs, both the graph structure and the set of conditional probability distributions can be learned from the available data.

Assuming a database of expression measurements for n genes in p consecutive time points, equally spaced over time, it is possible to indicate with $Y(t) = [Y_1(t), Y_2(t) \dots Y_n(t)]$ the set of random variables representing gene expression values at time t . In order to derive the DBN encoding the dependencies over the random variables Y in the different time points, it is assumed that the process under study (the dynamics of gene expression) is *Markovian*, that is, $p[Y(t+1)|Y(0) \dots Y(t)] = p[Y(t+1)|Y(t)]$ and *stationary*, that is, the transition probability $p[Y(t+1)|Y(t)]$ is independent of t .

Thanks to these assumptions, it is necessary to learn only the transition network between the variables at time t and at time $t+1$ [68]. To this aim, a probability model and a search strategy must be chosen.

Linear Gaussian networks suppose that the variables $Y_1 \dots Y_n$ are all continuous and that the conditional distribution of each variable Y_i given its parents, follows a *Gaussian distribution* with mean μ_i that is a linear function of the parent

variables [2]. The dependency of each variable on its parents is therefore represented by the linear regression equation

$$\mu_i = \beta_{i0} + \sum_j \beta_{ij} y_{ij} \quad 5.14$$

that models the conditional mean of Y_i at time $t + 1$ given the parent values y_{ij} , measured at time t .

Once the probability model has been chosen, learning the structure of the network can be approached as a model selection problem, which requires the choice of a scoring metric and a search strategy to explore the space of possible alternative models. The Bayesian solution consists in finding the network model with the maximum *posterior probability* given the data. This posterior is proportional to the *marginal likelihood* if it is assumed that all models are a priori equally likely. A significant advantage of using Gaussian distributions and linear dependencies of the children on their parents is that, when there is no missing data, the marginal likelihood can be calculated in closed form. The computation is therefore very efficient and the search process significantly speeded up. As it is not feasible to perform an exhaustive search over the space of all possible networks encoding the probabilistic dependencies among the n analyzed variables, it is possible to adapt the finite horizon local search proposed by Cooper and Herskovits [69] in order to explore the dependency of each variable Y_i on all the variables at the previous time point.

As stated before, the use of DBNs allows one to overcome the inability of Bayesian networks to represent cycles among variables and thus makes the discovery of feedback loops in gene networks feasible. Indeed, the necessary acyclic structure of the directed graph that encodes the dependencies between the network variables is no longer a limitation in the framework of DBNs. Considering, for example, two genes A and B and indicating with the subscripts t and $t + 1$ their expression values in two consecutive time points, if two links $A_t \rightarrow B_{t+1}$ and $B_t \rightarrow A_{t+1}$ are found through learning a DBN, it is possible to say that there is a feedback loop involving these two genes. Loops are more easily identified if the transition network inferred with the DBN algorithm is translated into a cyclic graph in which nodes referring to the same variable at consecutive time points are collapsed into a single node. An example is shown in Figure 5.9.

Recently Ferrazzi et al. have investigated the performance of Gaussian networks in modeling cellular systems [67]. In particular the following questions were addressed: is the proposed approach able to describe the complex dynamics of a cell? Is it able to infer the true underlying relationships among its state variables (gene expression/protein concentration values)? In order to have a benchmark data set on which to test the approach, data simulated through a set of nonlinear differential equations that describes the budding yeast cell cycle were exploited [70]. The whole model contains 36 differential equations; most of the variables represent protein concentrations, while others are auxiliary variables representing the mass and timing of cell cycle events. The dataset simulated in the case of wild-type cells was analyzed with the DBN approach described above.

Results showed that the model thus learned was able to effectively describe the dynamics of the analyzed system. Moreover, the “true parents” of each variable A (i.e., the other variables that appear in the differential equation describing A’s

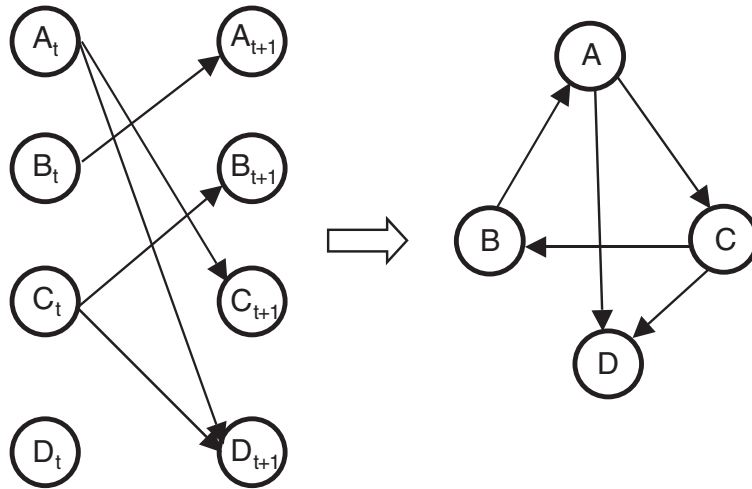


Figure 5.9 Example of translation of the transition network inferred by the DBN algorithm (*left*) into a cyclic graph (*right*). In this graph, feedback loops are more easily identified.

dynamics) were compared with the parents inferred through the DBN algorithm. The recall and the precision were then calculated: the recall corresponds to the fraction of “true parents” correctly inferred by the DBN algorithm, while the precision is the fraction of inferred parents that are also “true parents.” This accuracy calculation showed that the DBN algorithm was able to infer an interesting number of the “correct” relationships among the variables, as it provided results on average characterized by a 30% recall and an analogous precision.

The study confirmed that dynamic Bayesian networks can be effectively exploited to infer a synthetic description of the system under study, useful to guide researchers to further, deeper analyses. DBNs not only offer a phenomenological description of the dynamics of cellular systems, but also suggest hypotheses about the causal interactions among the cell’s state variables.

5.6 Conclusion

Systems biology can be defined as “the study of the behavior of complex biological organization and processes in terms of the molecular constituents” [1]. Systems biology thus doesn’t investigate the components of a cellular system one at a time but studies the properties of the whole system as they emerge from the complex interactions among them. This field has attracted the attention of the engineering community, as it was quickly recognized that engineering concepts and methods offer a natural and very effective framework to describe cellular systems.

As discussed in this chapter, the cell is a dynamical system that shares some basic control structures with engineered systems. Any dynamical system is characterized by a set of state variables, defined as the minimal set of variables able to summarize the past history of the system, so that the knowledge of their current value and of the inputs to the system is necessary and sufficient to determine the future output. Inputs, outputs, and state variables of the system are not absolute entities but are dependent on the analyzed processes and therefore on the level of

detail used in the model. A crucial step in systems biology studies is indeed the choice of the type of model to be used [71]. Kinetic models, for example, aim at representing the chemical reactions that take place during a certain process; in this case the state is defined by the concentrations of all the molecules involved in the reactions. Of course the model choice must be compatible with the type and amount of available data.

In this chapter the reverse engineering of gene networks from DNA microarray temporal data was discussed. These data do not allow the modeling of the true underlying set of transcription processes within a cell, as in these processes proteins and smaller molecules are also involved and should therefore be considered as state variables in the model. However, an abstract view of the cellular system, given by the network of connections among genes and by the associated model of the dynamics of gene expression, is also able to suggest potential functional relationships among the connected genes. It can even be useful to further increase the level of abstraction and, instead of modeling interactions among single genes, modeling groups of genes sharing similar properties (such as coexpressed genes) and then modeling the interactions among these groups. An approach in this direction was proposed by Segal et al., who developed a method, based on probabilistic graphical models, able to infer regulatory modules from gene expression data, that is, sets of “genes that are regulated in concert by a shared regulation program that governs their behavior” [54, 72].

An engineering perspective can help in choosing the right level of detail to study cellular systems: the description of the system dynamics by means of state variables, inputs, and outputs allows disregard of unnecessary details and capture of only the needed information.

A more ambitious goal is the design of reengineering interventions aimed at modifying cellular behavior. In this case systems theory can be used to simulate the effect of changes due to cell manipulations, while control theory can be exploited to properly select the different kinds of intervention strategies. Although a straightforward application of such approach is not possible, the basic principles of control engineering may provide guidelines to understand molecular dynamics and to act in order to obtain desirable properties. Reaching this goal would add a new crucial piece of knowledge to biomedical sciences.

Acknowledgments

We thank Felix B. Engel for discussions and helpful suggestions. Fulvia Ferrazzi was partially funded by the Consorzio Italia–MIT.

References

- [1] Kirschner, M. W., “The meaning of systems biology,” *Cell*, Vol. 121, May 20, 2005, pp. 503–504.
- [2] Sebastiani, P., M. Abad, and M. F. Ramoni, “Bayesian networks for genomic analysis,” in *Genomic Signal Processing and Statistics*, E. R. Dougherty et al. (eds.), New York: Hindawi, 2005, pp. 281–320.

- [3] Hartwell, L. H., et al., "From molecular to modular cell biology," *Nature*, Vol. 402, Dec. 2, 1999, pp. C47–C52.
- [4] Csete, M. E., and J. C. Doyle, "Reverse engineering of biological complexity," *Science*, Vol. 295, Mar. 1, 2002, pp. 1664–1669.
- [5] Sontag, E. D., "Molecular systems biology and control," *Eur. J. Control*, Vol. 11, 2005, pp. 396–435.
- [6] Khammash, M., and H. El-Samad, "Systems biology: from physiology to gene regulation," *IEEE Control Systems Magazine*, Vol. 24, Aug. 2004, pp. 62–76.
- [7] Sontag, E. D., "Some new directions in control theory inspired by systems biology," *Syst. Biol.*, Vol. 1, 2004, pp. 9–18.
- [8] Khoo, M. C. K., *Physiological Control Systems: Analysis, Simulation, and Estimation*, IEEE Press, 2000.
- [9] Bradley, E., "Analysis of Time Series," in *Intelligent Data Analysis: an Introduction*, 2nd ed., M. Berthold and D. J. Hand (eds.), New York: Springer, 2003, pp. 199–227.
- [10] Eddy, S. R., "What is a hidden Markov model?" *Nat. Biotechnol.*, Vol. 22, Oct. 2004, pp. 1315–1316.
- [11] Stelling, J., et al., "Robustness of cellular functions," *Cell*, Vol. 118, Sept. 17, 2004, pp. 675–685.
- [12] Lauffenburger, D. A., "Cell signaling pathways as control modules: complexity for simplicity?," *Proc. Natl. Acad. Sci. USA*, Vol. 97, May 9, 2000, pp. 5031–5033.
- [13] Morgan, D. O., "Cyclin-dependent kinases: engines, clocks, and microprocessors," *Annu. Rev. Cell Dev. Biol.*, Vol. 13, 1997, pp. 261–291.
- [14] Barkai, N., and S. Leibler, "Robustness in simple biochemical networks," *Nature*, Vol. 387, June 26, 1997, pp. 913–917.
- [15] Yi, T. M., et al., "Robust perfect adaptation in bacterial chemotaxis through integral feedback control," *Proc. Natl. Acad. Sci. USA*, Vol. 97, Apr. 25, 2000, pp. 4649–4653.
- [16] Becskei, A., and L. Serrano, "Engineering stability in gene networks by autoregulation," *Nature*, Vol. 405, June 1, 2000, pp. 590–593.
- [17] Hasty, J., D. McMillen, and J. J. Collins, "Engineered gene circuits," *Nature*, Vol. 420, Nov. 14, 2002, pp. 224–230.
- [18] El-Samad, H., et al., "Surviving heat shock: control strategies for robustness and performance," *Proc. Natl. Acad. Sci. USA*, Vol. 102, Feb. 22, 2005, pp. 2736–2741.
- [19] Tomlin, C. J., and J. D. Axelrod, "Understanding biology by reverse engineering the control," *Proc. Natl. Acad. Sci. USA*, Vol. 102, Mar. 22, 2005, pp. 4219–4220.
- [20] Angeli, D., and E. D. Sontag, "Monotone control systems," *IEEE Trans. on Automatic Control*, Vol. 48, 2003, pp. 1684–1698.
- [21] Kholodenko, B. N., et al., "Untangling the wires: a strategy to trace functional interactions in signaling and gene networks," *Proc. Natl. Acad. Sci. USA*, Vol. 99, Oct. 1, 2002, pp. 12841–12846.
- [22] Sontag, E., A. Kiyatkin, and B. N. Kholodenko, "Inferring dynamic architecture of cellular networks using time series of gene expression, protein and metabolite data," *Bioinformatics*, Vol. 20, Aug. 12, 2004, pp. 1877–1886.
- [23] Broman, K. W., and E. Feingold, "SNPs made routine," *Nature Methods*, Vol. 1, Nov. 2004, pp. 104–105.
- [24] Cai, W. W., et al., "Genome-wide detection of chromosomal imbalances in tumors using BAC microarrays," *Nat. Biotechnol.*, Vol. 20, Apr. 2002, pp. 393–396.
- [25] Sebastiani, P., E. Gussoni, and M. F. Ramoni, "Statistical challenges in functional genomics," *Statistical Sci.*, Vol. 18, 2003, pp. 33–70.
- [26] Kohane, I. S., A. T. Kho, and A. J. Butte, *Microarrays for an Integrative Genomics*, Cambridge, MA: MIT Press, 2003.

- [27] Hoheisel, J. D., "Microarray technology: beyond transcript profiling and genotype analysis," *Nat. Rev. Genet.*, Vol. 7, Mar. 2006, pp. 200–210.
- [28] Allison, D. B., et al., "Microarray data analysis: from disarray to consolidation and consensus," *Nat. Rev. Genet.*, Vol. 7, Jan. 2006, pp. 55–65.
- [29] Schena, M., et al., "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, Vol. 270, Oct. 20, 1995, pp. 467–470.
- [30] Lockhart, D. J., et al., "Expression monitoring by hybridization to high-density oligonucleotide arrays," *Nat. Biotechnol.*, Vol. 14, Dec. 1996, pp. 1675–1680.
- [31] Duggan, D. J., et al., "Expression profiling using cDNA microarrays," *Nat. Genet.*, Vol. 21, Jan. 1999, pp. 10–14.
- [32] Lipshutz, R. J., et al., "High density synthetic oligonucleotide arrays," *Nat. Genet.*, Vol. 21, Jan. 1999, pp. 20–24.
- [33] Golub, T. R., et al., "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, Vol. 286, Oct. 15, 1999, pp. 531–537.
- [34] Slonim, D., et al., "Class prediction and discovery using gene expression data," in *Fourth Ann. Int. Conf. on Comput. Mol. Biol. (RECOMB)*, 2000, pp. 263–272.
- [35] Domon, B., and R. Aebersold, "Mass spectrometry and protein analysis," *Science*, Vol. 312, Apr. 14, 2006, pp. 212–217.
- [36] Hartemink, A. J., et al., "Combining location and expression data for principled discovery of genetic regulatory network models," *Pacific Symp. Biocomput.*, 2002, pp. 437–449.
- [37] Bernard, A., and A. J. Hartemink, "Informative structure priors: joint learning of dynamic regulatory networks from multiple types of data," *Pacific Symp. Biocomput.*, 2005, pp. 459–470.
- [38] Segal, E., R. Yelensky, and D. Koller, "Genome-wide discovery of transcriptional modules from DNA sequence and gene expression," *Bioinformatics*, Vol. 19, Suppl. 1, 2003, pp. i273–i282.
- [39] Segal, E., H. Wang, and D. Koller, "Discovering molecular pathways from protein interaction and gene expression data," *Bioinformatics*, Vol. 19, Suppl. 1, 2003, pp. i264–i271.
- [40] Nariai, N., et al., "Estimating gene regulatory networks and protein-protein interactions of *Saccharomyces cerevisiae* from multiple genome-wide data," *Bioinformatics*, Vol. 21, Suppl. 2, Sept. 1, 2005, pp. ii206–ii212.
- [41] Brazhnik, P., A. de la Fuente, and P. Mendes, "Gene networks: how to put the function in genomics," *Trends Biotechnol.*, Vol. 20, Nov. 2002, pp. 467–472.
- [42] Ramoni, M. F., P. Sebastiani, and I. S. Kohane, "Cluster analysis of gene expression dynamics," *Proc. Natl. Acad. Sci. USA*, Vol. 99, July 9, 2002, pp. 9121–9126.
- [43] de Jong, H., "Modeling and simulation of genetic regulatory systems: a literature review," *J. Comput. Biol.*, Vol. 9, 2002, pp. 67–103.
- [44] Butte, A. J., et al., "Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks," *Proc. Natl. Acad. Sci. USA*, Vol. 97, Oct. 24, 2000, pp. 12182–12186.
- [45] D'Haeseleer, P., S. Liang, and R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering," *Bioinformatics*, Vol. 16, Aug 2000, pp. 707–726.
- [46] Liang, S., S. Fuhrman, and R. Somogyi, "Reveal, a general reverse engineering algorithm for inference of genetic network architectures," *Pacific Symp. Biocomput.*, 1998, pp. 18–29.
- [47] Akutsu, T., S. Miyano, and S. Kuhara, "Identification of genetic networks from a small number of gene expression patterns under the Boolean network model," *Pacific Symp. Biocomput.*, 1999, pp. 17–28.
- [48] Shmulevich, I., et al., "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, Vol. 18, Feb. 2002, pp. 261–274.

- [49] Shmulevich, I., E. R. Dougherty, and W. Zhang, "Gene perturbation and intervention in probabilistic Boolean networks," *Bioinformatics*, Vol. 18, Oct. 2002, pp. 1319–1331.
- [50] Choudhary, A., et al., "Intervention in a family of Boolean networks," *Bioinformatics*, Vol. 22, Jan. 15, 2006, pp. 226–232.
- [51] Friedman, N., et al., "Using Bayesian networks to analyze expression data," *J. Comput. Biol.*, Vol. 7, 2000, pp. 601–620.
- [52] Pe'er, D., et al., "Inferring subnetworks from perturbed expression profiles," *Bioinformatics*, Vol. 17, Suppl. 1, 2001, pp. S215–224.
- [53] Segal, E., et al., "Rich probabilistic models for gene expression," *Bioinformatics*, Vol. 17, Suppl. 1, 2001, pp. S243–S252.
- [54] Segal, E., et al., "Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data," *Nat. Genet.*, Vol. 34, June 2003, pp. 166–176.
- [55] Jansen, R., et al., "A Bayesian networks approach for predicting protein-protein interactions from genomic data," *Science*, Vol. 302, Oct. 17, 2003, pp. 449–453.
- [56] Cai, Z., et al., "Bayesian approach to discovering pathogenic SNPs in conserved protein domains," *Hum. Mutat.*, Vol. 24, Aug. 2004, pp. 178–184.
- [57] Sebastiani, P., et al., "Genetic dissection and prognostic modeling of overt stroke in sickle cell anemia," *Nat. Genet.*, Vol. 37, Apr. 2005, pp. 435–440.
- [58] Murphy, K., and S. Mian, *Modelling Gene Expression Data Using Dynamic Bayesian Networks*, Computer Science Division, Univ. Calif., Berkeley, 1999.
- [59] Ong, I. M., J. D. Glasner, and D. Page, "Modelling regulatory pathways in *E. coli* from time series expression profiles," *Bioinformatics*, Vol. 18, Suppl. 1, 2002, pp. S241–S248.
- [60] Perrin, B. E., et al., "Gene networks inference using dynamic Bayesian networks," *Bioinformatics*, Vol. 19, Suppl. 2, 2003, pp. II138–II148.
- [61] Kim, S., S. Imoto, and S. Miyano, "Inferring gene networks from time series microarray data using dynamic Bayesian networks," *Brief. Bioinform.*, Vol. 4, 2003, pp. 228–235.
- [62] Kim, S., S. Imoto, and S. Miyano, "Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data," *Biosystems*, Vol. 75, July 2004, pp. 57–65.
- [63] Rangel, C., et al., "Modeling T-cell activation using gene expression profiling and state-space models," *Bioinformatics*, Vol. 20, June 12, 2004, pp. 1361–1372.
- [64] Husmeier, D., "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks," *Bioinformatics*, Vol. 19, Nov. 22, 2003, pp. 2271–2282.
- [65] Yu, J., et al., "Advances to Bayesian network inference for generating causal networks from observational biological data," *Bioinformatics*, Vol. 20, Dec. 12, 2004, pp. 3594–3603.
- [66] Dojer, N., et al., "Applying dynamic Bayesian networks to perturbed gene expression data," *BMC Bioinformatics*, Vol. 7, 2006, p. 249.
- [67] Ferrazzi, F., et al., "Dynamic Bayesian networks in modelling cellular systems: a critical appraisal on simulated data," *19th IEEE Symp. on Computer-Based Medical Systems*, 2006, pp. 544–549.
- [68] Friedman, N., K. Murphy, and S. Russel, "Learning the structure of dynamic probabilistic networks," in *Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 139–147.
- [69] Cooper, G. F., and E. Herskovitz, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, Vol. 9, 1992, pp. 309–347.
- [70] Chen, K. C., et al., "Integrative analysis of cell cycle control in budding yeast," *Mol. Biol. Cell.*, Vol. 15, Aug. 2004, pp. 3841–3862.
- [71] Ideker, T., T. Galitski, and L. Hood, "A new approach to decoding life: systems biology," *Annu. Rev. Genomics Hum. Genet.*, Vol. 2, 2001, pp. 343–372.
- [72] Segal, E., et al., "Learning module networks," *J. Machine Learning Res.*, Vol. 6, 2005, pp. 557–588.

Modeling Cellular Networks

Tae Jun Lee, Dennis Tu, Chee Meng Tan,
and Lingchong You

6.1 Introduction

Systems-level understanding of cellular dynamics is important for identifying biological principles and may serve as a critical foundation for developing therapeutic strategies. To date, numerous developments of therapeutics have been based on identification and comprehensive analysis of cellular dynamics, especially in the involved pathways. In cancer therapy, for instance, many researchers have focused on oncogenic pathways such as the Rb pathway, whose in-depth understanding of the pathway dynamics promises effective therapeutics [1–7]. The effectiveness of this approach in the development of cancer therapeutics has been illustrated in in vivo pre-clinical tests of the engineered adenovirus ONYX-015 and ONYX-411. These adenoviruses, engineered to target mutations in the Rb or p53 pathway for killing, have demonstrated high selectivity and efficiency in viral replication in tumor cells for cell killing [8, 9]. However, clinical application of these methods is hindered by lack of ability to precisely predict and regulate cellular responses. This ability is essential in minimizing complications and side effects. Especially, a large amount of biology data on these pathways generated by rapid advancements in biotechnologies and molecular biology renders integrated understanding of the pathway dynamics impossible by intuition alone. Therefore, a more systematic approach allowing incorporation of the multitude of information is necessary to improve prediction and regulation of cellular responses.

To this end, mathematical modeling is becoming increasingly indispensable for basic and applied biological research. Essentially, a mathematical model is a systematic representation of biological systems, whose analysis can confer quantitative predicting power. In recent years, advanced computing power combined with improved numerical methods have made it possible to simulate and analyze dynamics of complex cellular networks [10–19].

Mathematical modeling is useful in a number of ways. One of the common applications of mathematical modeling is to analyze cellular networks systematically. For example, although the mitogen-activated protein kinase (MAPK) was known to control multiple cellular responses such as cell growth, survival, or differentiation, the molecular mechanisms for these divergent behaviors were not fully elucidated.

Consequently, several models on the MAPK pathway have been developed that differentiate activation patterns in response to epidermal growth factors and neural growth factors [20], characterize the signal-response relationship [21, 22], and suggest the significance of feedback control in complete signal adaptation [23]. A more extensive modeling work investigates the emergent properties that arise from multiple signaling pathways [24]. These works illustrate the utility of mathematical modeling in understanding complex biological systems that intuition alone cannot handle.

Another use of mathematical modeling has been demonstrated in devising strategies to control cellular dynamics. The concentrations of MAPK phosphatase have been shown to play a key role in whether the MAPK pathway demonstrates monostable or bistable states [22]. Sasagawa and colleagues used their MAPK model to identify “critical nodes,” in which perturbations resulted in dramatic changes in system behaviors [20]. A number of critical nodes that are responsible for diverse cellular actions have also been suggested in the insulin-signaling pathways based on biochemical and computational data [25]. Such characterization of input-output response or identification of critical nodes can be utilized to effectively modulate cellular dynamics.

Furthermore, modeling can form a basis for the development of therapeutics for medical applications. Various pathway models including the MAPK models described above can be useful in designing, or evaluating the effectiveness of, therapeutic drugs *in silico* [20–24]. The predictive power and therapeutics design principles that these models offer can facilitate development of therapeutics [26–28]. Stemming from these studies on the MAPK signaling pathways, Kitano and colleagues have developed an epidermal growth factor receptor (EGFR) pathway map in a software that is shared and compatible with other simulation and analysis packages [29]. Such efforts to make available and share information on biological pathways among researchers exemplify an inclination towards understanding of biology via mathematical modeling.

Despite advantages of mathematical modeling for basic and applied biological research, there remain many challenges in constructing and analyzing models. Modeling of biological systems is always accompanied by assumptions, which are predicated on the modeler’s goals. Therefore, a successful modeling work requires clear justification of these assumptions. Even with clear, justified goals, a modeler is faced with another challenge: lack of detailed, quantitative biological information. While biotechnologies continue to advance our knowledge of the building blocks of biological systems, parameters for the kinetics of interactions among them are often unknown. Various methodologies for inferring reaction mechanisms and parameters have been proposed [30–36]. Yet high-throughput biological data, generated by microarray experiments or protein expression profiling, are often not of sufficiently high resolution for using these techniques. To address these issues, a combination of mathematical modeling and experimental validations is required. Iterations of model construction, system analysis, and experimental validation improve accuracy of the model and lead to increased predictive power. In particular, the power to quantify gene expression with high temporal resolution at the population level or single cell level will likely complement high-throughput technologies in facilitating inference of reaction mechanisms and parameters [37–43].

In this chapter, we present methodologies on modeling, simulation, and analysis of natural and synthetic cellular networks. We note that different types of mathematical models are widely used. Here we limit our scope to kinetic models, which represent systems of interest as coupled chemical reactions. By so doing, we steer away from discussing other widely used mathematical models, such as Boolean models and those focusing on spatial dynamics. We illustrate construction of mathematical models of cellular networks with increasing complexity. Further, we highlight mathematical representations commonly used to describe such cellular networks and discuss common techniques for analyzing modeling results. Importantly, we show how to relate modeling results to real biological systems and to make predictions that can be validated by experiments. We use relatively simple, well-characterized systems to illustrate these processes.

6.2 Construction and Analysis of Kinetic Models

Construction of a kinetic model can be a daunting task for a system consisting of a large number of components with complex interactions. To build an experimentally tractable model, it is important to define the scope of abstraction. Once the scope is defined, a conventional approach begins with a minimal diagram that includes key components and interactions among them. Identification of the key components and interactions is based on the current knowledge of biology and frequently on intuition and experience. Depending on the focus of study, a modeler may choose to emphasize certain signaling pathways while deemphasizing less relevant ones. These processes often accompany “lumping” or deletion of molecular interactions or components. Once the diagram is completed, a minimal mathematical model is constructed from the information embedded in the diagram and is further refined or extended to reflect new hypotheses or experimental measurements. Simulation of the final model reveals the network dynamics, which in turn gives insights into the intrinsic design principles.

6.2.1 Parameter Estimation and Modeling Resources

A major challenge in model formulation is determination of reaction mechanisms and estimation of parameters. In some systems, the network behaviors are defined mostly by the architecture of the system. These systems are highly robust to a wide range of parameters. In others, system dynamics are determined not only by the architecture, but also the parameters, which are often poorly understood. Therefore, construction of a meaningful mathematical model of a biological pathway requires two critical elements: interactions between molecular species and the kinetics of the interactions.

As the first step, we need to know the interactions between the molecular species in the model. Several pathway databases are available for this purpose: EcoCyc [44], Kegg [45], ERGO [46], aMAZE [47], ExPASy [48], www.sbml.org, STKE (Sci), and Nature Signaling update (www.signaling-gateway.org). The pathways included in these databases are retrieved and constructed from specialized databases such as GenBank, PDB, and EMBL. These pathway databases often

provide detailed information on the molecular species. Next, we need to determine the kinetics of the interactions. In most cases, kinetic parameters are obtained from the literature data. Alternatively, we can use kinetic parameters from typical values, which can be based on values inferred from related process or even the experience of the modeler.

For every biological system, model construction usually goes through iterations of model construction, experimental validation, and model refinement (in terms of reaction mechanisms or parameter values) (Figure 6.1). These steps will be repeated until the mathematical model matches the experimental data to a satisfactory degree. This process can be considered as a special case of “reverse engineering” biological pathways. Additional methods, such as Bayesian [32], maximum likelihood [36], and genetic algorithms [33] can be used to infer more qualitative connectivity of biological networks from high-throughput experimental data.

6.2.2 A Modular Approach to Model Formulation

Modeling and analysis of complex biological systems may benefit from a modular approach, in which a biological system is conceptualized as a combination of smaller subnetworks with well-recognizable functions, termed motifs and modules [49, 50]. The distinction between motifs and modules is often based on the size difference but is not always clear-cut. We here use the two terms interchangeably. That is, we consider all small, conserved regulatory subnetworks as “modules,” classifiable on the basis of function, architecture, dynamics, and biochemical process. Such conceptualization may provide insight into the qualitative network dynamics at the systems level, and it helps clarify the modeling objective and generate qualitative hypotheses. In addition, it forms the basis for incorporating mathematical equations, with which more quantitative understanding can be attempted.

The dynamics of a module are governed by both network connectivity and associated parameter values. In general, increasing complexity in either variables or connectivity will result in more complex dynamics, and modules with feedback control may show properties that are difficult to grasp by intuition alone. Structures and properties of some well-defined feedback control modules are shown in Table 6.1, where we have summarized their key properties. For example, a module with one variable demonstrates either monostable or bistable properties with negative or positive feedback control, respectively, but it is impossible to generate oscillations

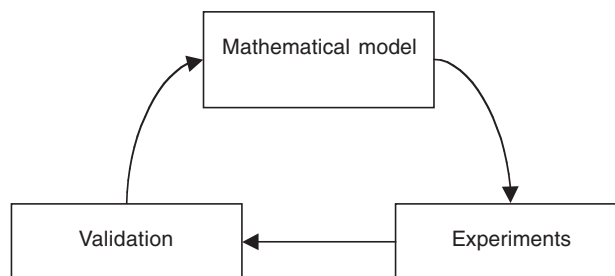


Figure 6.1 Refining models of biological networks. Iteration of model construction and experiments enable parameter and kinetics estimation and model refinement. The experimental data can be matched to the model with various computational methods.

Table 6.1 Well-defined feedback modules involving negative, positive, or both types of regulation. In general, an increasing number of variables and more complex connectivity leads to richer dynamics.

Number of variables	Negative regulation only	Negative and positive regulation
1	 Monostable	 Bistable
2	 Bistable	 Oscillation
3	 Oscillation	 Oscillation ◦ ◦ Chaos

with a single variable in the absence of time delay. Monostable, bistable, or oscillatory behaviors, but not chaos, can be generated with a two-variable module. Modules with higher number of variables can demonstrate much richer dynamics such as chaos (Table 6.1) [49, 51, 52].

Various feedback control mechanisms confer properties useful for different biological functions. For example, negative feedback control is essential in homeostasis, a process of maintaining the system’s internal environment in a steady state. Without feedback control, sudden external changes such as those in temperature or salinity may induce significant internal damages that can be fatal to a cell. Negative feedback control can buffer the impact of such changes and facilitate homeostasis [53]. In attempts to engineer gene circuits, this property has been used to reduce variations in gene expression [54]. In addition, negative feedback may increase the response speed of simple gene cascades [55].

Positive feedback can create bistable behaviors. The synthetic biology approach has been used to develop bistable switches whose overall molecular mechanism is based on autocatalysis of a single gene [56, 57]. These networks may be considered as synthetic models of their natural counterparts, such as signaling networks that control cell cycle regulation [58, 59] and regulation of the lac operon [60]. Bistable switches can also be realized in a positive feedback system by combining negative regulations. A recent study on a synthetic “toggle” switch, a two-component module in which two transcriptional repressors negatively regulate each other, is shown to achieve bistable switching behaviors [61]. A combination of negative or positive regulation between two or more components can give rise to oscillations. This was theoretically or experimentally characterized in *Escherichia coli* [62–65].

In addition to monostable, bistable, or oscillatory modules, network architectures with other connectivity have also been identified, and their properties and biological significance have been characterized [49, 52, 66, 67]. Importantly, these modules

often maintain similar functions across different species. For example, oscillator modules are the molecular mechanisms that underlie molecular, physiological, and behavioral rhythms [68, 69] or pattern formations [70], and bistability modules may govern the cell's entry into the cell cycle and be responsible for controlling cell differentiation [58, 71–74]. Thus, thorough analysis of a module in one context can provide insight into its functional roles under a wide spectrum of conditions.

6.2.3 Basic Kinetics

In kinetic modeling, a biological system is considered to be a series of chemical reactions, whose kinetics can be described by rate expressions. The system is often composed of multiple reactions, which occur through direct interactions among reactants. If these interactions are elementary reactions, their rates can be modeled following the mass action law. That is, the reaction rate is proportional to the product of reactant concentrations. However, most biological models are frequently formulated as consisting of more complex reaction mechanisms. One important class is enzyme-catalyzed reactions, which are critical for live systems where virtually all reactions are too slow to support life without enzymes. The enzymes provide a way to regulate reactions at appropriate rates and conditions.

A commonly used reaction model for enzymatic reactions is the Michaelis-Menten equation. In this reaction mechanism, one assumes that the enzyme is not consumed and the total concentration of enzyme stays constant. It only interacts directly with the substrate to form an enzyme-substrate complex, which leads to the synthesis of the product:



Assuming that the intermediate (ES) is at the quasi-steady-state and the substrate is in excess, we can derive the Michaelis-Menten equation:

$$\frac{dP}{dt} = \frac{V_{\max}[S]}{K_M + [S]} \quad 6.2$$

where $V_{\max}$ is the maximal reaction rate ($k_2[E]_{\text{Total}}$, where $[E]_{\text{Total}}$ is the total enzyme concentration) and K_M is the Michaelis-Menten constant $\frac{(k_r + k_2)}{k_f}$.

Another recurring scheme in modeling cellular networks is the representation of gene expression. Expression of a single gene involves two basic steps: transcription and translation. This simplistic view of gene regulation starts with transcription, where the RNA polymerase binds the promoter of a gene to result in mRNA synthesis. The mRNA that carries coded information binds with ribosome, and the coded information is translated into protein [Figure 6.2(a)].

In real systems, gene expression can be regulated at multiple layers involving interactions among inducers, repressors, and operator sites. The interactions of these components lead to two general categories of transcriptional regulations: activation and repression. When an activator binds to the operator site, this complex leads to recruitment of RNA polymerase (RNAP) and synthesis of mRNA (Figure 6.2b). In contrast, binding of a repressor will prevent initiation of transcription by blocking

the RNAP. In the absence of cooperative interactions, such as dimerization and synergistic binding of transcription regulators to promoters, both types of regulation can be mathematically described by using Michaelis-Menten type of kinetics [Figure 6.2b, c)].

If the transcription regulator acts as a dimer or multimer, and/or if it binds synergistically to multiple operator sites, transcription regulation can be modeled by higher-order expressions, such as the commonly used Hill kinetics:

$$\frac{dP}{dt} = \frac{V_{\max}[S]^n}{K_M^n + [S]^n} \quad 6.3$$

where n is called the Hill coefficient. For $n = 1$, Hill kinetics is the same as the Michaelis-Menten kinetics. However, for the response curve that has a different slope from what is predicted by Michaelis-Menten kinetics, n can be adjusted to fit the Hill kinetics curve. Detailed treatment of this can be found in [75, 76].

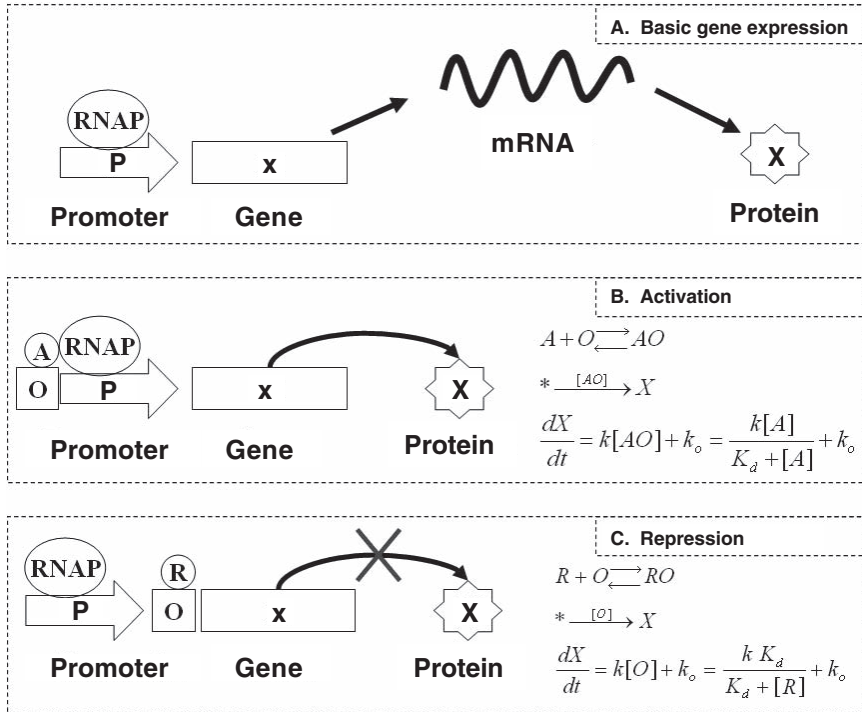


Figure 6.2 Modeling gene regulation. A simplified view of gene regulation is shown in (a). Initiation of mRNA synthesis can be triggered by either transcription activation (b) or transcription repression (c), which can be mathematically represented with the Michaelis-Menten type of kinetics. In transcription activation, the synthesis of mRNA depends on the amount of activators (A) bound to the operator (O). In contrast, the production of mRNA is repressed when the operator is bound to the repressor (R). In this case, the rate of transcription can be assumed to be proportional to the concentration of free operator sites. We assume that the RNA polymerase is not rate limiting and that translation and transcription steps are lumped together with an overall synthesis rate constant k . k_o is the basal protein synthesis rate constant, and K_d is the dissociation constant for the binding between A or R and O.

6.2.4 Deterministic Models

By treating each interaction as a chemical reaction, one can account for the production and depletion of each species by using an ordinary differential equation (ODE). A coupled system of ODEs that describes the dynamics of all elements in the network constitutes an integrated kinetic model. The general form of ODE systems can be written as:

$$\begin{aligned}\frac{dx_1}{dt} &= f_1(x_1, x_2, x_3, \dots, x_n) \\ &\vdots \\ \frac{dx_n}{dt} &= f_n(x_1, x_2, x_3, \dots, x_n)\end{aligned}\tag{6.4}$$

where $x_1, x_2, \dots, x_n$ represent levels of different interacting species, and $f_1, f_2, \dots, f_n$ represent their corresponding rate expressions.

This representation often implies that the system dynamics occur in a well-stirred reactor in which bulk concentrations of the components are considered. Except for simple systems, an ODE-based kinetic model is often solved numerically using established methods [77–80]. Given the same parameter values, initial conditions, and simulation settings (e.g., error tolerance), different rounds of simulations will generate exactly the same temporal dynamics for each individual component. As such, an ODE model is also called a “deterministic” model. To assist in computational modeling research, a wide range of computational methods and tools have been developed for ODE systems [18, 81–86].

6.2.5 Cellular Noise and Stochastic Methods

ODE-based models are widely used to model dynamics of both natural and synthetic biological networks. For example, deterministic simulations predicted that a synthetic gene circuit of transcriptional repressors would cause sustained oscillations in gene expression [63]. Aspects of these predictions were verified in experiments where individual cells carrying the circuit displayed oscillatory behavior. However, the real system dynamics were quite stochastic when compared to deterministic simulation results from an ODE model. Specifically, oscillations in the repressilator occurred in only ~40% of individual cell lineages and were often out of phase with each other.

Such stochastic behavior is partially due to the intrinsically stochastic biochemical reactions among small numbers of molecules. These fluctuations in gene expression are often termed “noise” [87, 88]. In general, sources of noise include fluctuations in cellular components [40], transmitted noise from upstream genes [41], and other cellular processes unaccounted for by the model. Recently, the origin and propagation of noise in gene expression have been of central interest in many experimental studies [39, 43, 89–92].

The presence of cellular noise presents both a challenge and an opportunity for cellular function. On one hand, reliable function in the presence of noise requires strategies that reduce the impact of noise [89, 93]. For instance, one such mechanism that regulates noise is negative feedback, where the output from a system re-

duces its own output. In a biological context, this occurs when a protein inhibits its own expression by binding to its promoter. This mechanism has been shown to reduce noise in gene expression [54, 94]. On the other hand, noise may be beneficial by serving as a source for generating phenotypic diversity [93, 95], which can facilitate adaptation to changing environments or trigger cell differentiation [96].

Because of the important implications of noise for both natural and synthetic cellular networks, it is often useful to model stochastic dynamics. For a well-stirred, spatially homogeneous system, its stochastic temporal dynamics can be captured by a generic chemical master equation (CME) [97]:

$$\frac{\partial P(\tilde{x}, t | \tilde{x}_0, t_0)}{\partial t} = \sum_{j=1}^M a_j(\tilde{x} - \nu_j) \times P(\tilde{x} - \nu_j | \tilde{x}_0, t_0) - a_j(\tilde{x}) \times P(\tilde{x}, t | \tilde{x}_0, t_0) \quad 6.5$$

The first term of the equation describes probability of a species reacting at time t , while the second term describes the probability of a species remaining in its current state. $\tilde{x}$ is a vector containing the number of molecules for each species. $P(\tilde{x}, t)$ gives the probability of the system in state $\tilde{x}$ at time t . a_j is the propensity value of reaction j . ν_j is a vector containing the changes in state $\tilde{x}$ caused by reaction j . $\tilde{x}_0$ and t_0 are the initial state and time respectively.

One can solve the CME analytically only for very simple systems. As the system size increases beyond a few reactions, the analytical solution of the CME becomes intractable. When the number of reactions and the number of molecules increase, the number of possible paths increases exponentially. Gillespie proposed a simple algorithm to solve the CME numerically using a Monte-Carlo method [98]. In this formulation, each reaction is assumed to be an elementary reaction, where collisions between reactant molecules directly lead to formation of products. The probability that a reaction happens is dependent on its reaction propensity (Table 6.2), which is analogous to a rate expression in ODE-based models.

The reaction propensity describes the probability of one molecule colliding with another molecule, which leads to the firing of a chemical reaction. Note that the reaction propensity for dimerization reactions is equal to $c x_A (x_A - 1)/2$ rather than $c x_A x_A$ because a molecule cannot react with itself. This presents a consistent interpretation of stochastic rate constants which are normally calculated from conventional rate constants [98]. Given the reaction propensity, we can now define the state of a species at time t . In order to follow the evolution of the states through time, we have to calculate which reaction (μ) is firing at time t and how much time (τ) the reaction requires. The probability of the firing event is shown in (6.6) and it can be calculated by using the schemes illustrated in Table 6.3.

$$P(\tau, \mu) = a_j \times \exp(-a_0 \times \tau) \quad 6.6$$

Table 6.2 Reaction propensity for stochastic methods. The reaction propensity describes probability of one molecule colliding with another.

Reaction	Propensity
$A \rightarrow B$	$c x_A$
$A + B \rightarrow C$	$c x_A x_B$
$2A \rightarrow B$	$c x_A (x_A - 1)/2$

Table 6.3 Pseudocode of Gillespie algorithm adapted from [98].

-
1. Calculate $a_0 = \sum_j^M a_j$
 2. Generate two random numbers r_1 and r_2 from the uniform distribution (0, 1)
 3. Compute $\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$
 4. Compute μ that satisfies $\sum_j^\mu a_j \geq r_2 \times a_0 \geq \sum_j^{\mu-1} a_j$
 5. Execute reaction μ and advance time t by τ
-

Despite its simplicity, the computational cost of the Gillespie algorithm increases drastically with the number of reactions and the number of molecules in a system. The increment in computational cost is primarily due to the generation of random numbers (Step 2 in Table 6.3) and the enumeration of reactions to determine the next reaction (Step 4 in Table 6.3). For example, when the number of molecules is equal to 1×10^6 , τ will become excessively small (on the order of 1×10^{-6}), which then increases the number of time steps.

In order to simulate large-scale stochastic models, Gibson [99] proposed the “next reaction method” to improve computational efficiency of the Gillespie algorithm. The first improvement involves implementation of a tree data structure to store the reaction time of each reaction, which minimizes enumeration of the reactions at every time step. The second improvement uses a map data structure to minimize recalculation of the reaction propensity at every time step. The Gibson algorithm is significantly faster than the Gillespie algorithm for systems consisting of many reactions and many reacting species. It is also an exact algorithm in the sense that it satisfies the same basic assumptions as required by the Gillespie algorithm.

Several other algorithms were also proposed to improve computational speed of stochastic simulations. These algorithms are not exact and require users to pre-determine an extra parameter that affects accuracy of the numerical solutions. Tau-leap algorithms [100] predict multiple firing of fast reactions and, hence, reduce the total number of time steps. Another class of algorithms is a hybrid algorithm [93, 101], which models fast reactions subsets using either ODEs or Langevin equations (see below), while treating slow reaction subsets with the stochastic algorithms.

An alternative, widely used stochastic method remains in the framework of differential equations by adding an appropriate noise term to each of the ODEs that describe the biological network. The resulting stochastic differential equations (SDEs) can then be solved numerically. Different formulations of SDEs can be established for different types of simulation applications. With appropriate assumptions, one can obtain a special type of SDEs, the chemical Langevin equation [102], which has been used to model a variety of cellular networks.

$$\frac{dX_i(t)}{dt} = \sum_{j=1}^M v_{ji} a_j [X(t)] + \sum_{j=1}^M v_{ji} a_j^{1/2} [X(t)] \Gamma_j(t) \quad 6.7$$

where $X_i(t)$ is the number of molecules of a molecular species in the system at time t and i refers to the specific molecular species ($i = 1, \dots, N$). $X(t) \equiv [X_1(t), \dots, X_N(t)]$ is the state of the entire system at time t , $a_j[X(t)]$ is the rate for a specific reaction or molecular interaction ($j = 1, \dots, M$); ν_{ji} is a matrix describing the change in the number of molecules as a result of one molecular interaction. In other words, interactions that result in the synthesis of $X_i(t)$ are added and interactions that result in the degradation of $X_i(t)$ are subtracted; $\Gamma_j(t)$ are temporally uncorrelated, statistically independent Gaussian white noises.

SDEs are attractive in that they are computationally more efficient than the Gillespie algorithm and its derivatives. Also, by remaining in the framework of differential equations, they can facilitate in-depth analysis of system dynamics without always resorting to numerical simulations [103].

Regardless of the exact formulation of a stochastic algorithm, repeated rounds of stochastic simulations will generate different temporal dynamics for each individual species. One often uses an ensemble of simulated time courses to gain insights into noise characteristics, as well as how they are impacted by regulatory mechanisms. One way of quantifying noise in gene expression is to normalize the standard deviation of protein level with respect to the average protein level ($\gamma = \sigma/\mu$), where σ is the standard deviation of protein level and μ is the mean of protein level [40]. While this metric is direct and intuitive, some noise characteristics may be obscured by the more dominant small-number effects [89]. This may make it difficult to compare the noise of proteins that are being expressed at different levels. In this case, a more advantageous metric of quantifying noise is noise strength, or the variance of the protein level normalized with respect to the average protein level, $\zeta = \sigma^2/\mu$. Since gene expression is often controlled through transcription factors, noise levels can be compared among different genes regardless of their expression levels. This metric was recently used to analyze the relative contribution of transcription rates and translation rates to the noise characteristics of final protein products [103].

6.2.6 System Analysis Techniques

Given an integrated model, one can characterize the system behaviors using various analysis techniques, such as parametric sensitivity analysis and bifurcation analysis. These techniques allow for exploration of potential system dynamics and provide quantitative insights into emergent system behaviors, such as robustness. Such information is useful for revealing “design principles” of natural biological systems or guiding design and implementation of synthetic gene circuits.

6.2.6.1 Parametric Sensitivity Analysis

Sensitivity analysis is used to quantify changes in system behaviors in response to parameter changes. Different parameters may have varying impacts on the system dynamics and the degree of the impact can be quantified by a sensitivity value. A general method for computing the sensitivity value for an ODE system is

$$s(I; \phi_j) = \frac{\partial I}{\partial \phi_j} = \lim_{\Delta \phi_j \rightarrow 0} \frac{I(\phi_j + \Delta \phi_j) - I(\phi_j)}{\Delta \phi_j} \quad 6.8$$

where the sensitivity value is the ratio of change in the objective function of interest (I) to change in a parameter (ϕ_i).

Alternatively, the normalized form of sensitivity can be defined:

$$S(I; \phi_i) = \frac{\phi_i}{I} \times \frac{\partial I}{\partial \phi_i} = \frac{\partial \ln I}{\partial \ln \phi_i} = \frac{\phi_i}{I} \times s(I; \phi_i) \quad 6.9$$

This is also called logarithmic sensitivity. It is commonly used in metabolic control analysis [104] and has the feature of being dimensionless.

The objective function of interest is determined by the goals of the analysis. In the enzymatic synthesis of product that follows Michaelis-Menten kinetics, one may be interested in the change in the synthesis rate or in the steady-state product concentration as the Michaelis-Menten constant is varied. Therefore, there may be more than one sensitivity value for a given parameter. For an extensive treatment of sensitivity analysis, refer to [105].

Sensitivity analysis has been widely used in quantifying robustness of complex biological systems with respect to parametric perturbations [106–110]. In a complex system with a large number of parameters, the system behaviors may be robust to changes in various parameters. Especially, feedback controls and backup or compensation mechanisms in biological systems confer additional layers of robustness [14, 111–113]. Accurate identification of the underlying mechanisms for such robustness is challenging, since the system behaviors result from both parameters and system architecture. By distinguishing the impact of parameters from that of the architecture, sensitivity analysis provides a way to characterize system robustness. Such mathematical exploration of various system behaviors may serve as a guide in realizing system behaviors as desired experimentally. Specifically, if the parameters with high sensitivity values can be controlled, higher efficiency in biologically feasible experiment designs and data analysis can be achieved.

6.2.6.2 Bifurcation Analysis

While sensitivity analysis provides a quantitative measure of the dependence of system dynamics on parameters, bifurcation analysis focuses on a qualitative understanding of the system dynamics. Similar to sensitivity analysis, bifurcation analysis monitors changes in system behaviors in response to parameter changes, except that the goal is to explore *qualitative* changes in the systems dynamics. Bifurcation analysis is performed by varying a parameter until a qualitative change in dynamics is observed. The value at which this occurs is called the bifurcation point.

A quantitative measure of the stability can be achieved by a simple analytical method called linear stability analysis. This method provides a numerical value for the rate of decay to the stable steady-state solution from a small perturbation. Let us consider a model consisting of only one species,

$$\frac{dx}{dt} = f(x)$$

Linear stability analysis begins with steady-state solutions (x_s), which can be found by equating the right-hand side of the ODE expression to 0 and solving for the

species concentration of interest. Adding a small perturbation, $x = x_s + \delta(t)$, the right-hand side becomes

$$f(x) = f[x_s + \delta(t)] \xrightarrow{\text{Taylor's expansion}} f(x_s) + \delta(t)f'(x_s) + O[\delta(t)]^2$$

Assuming that the higher order terms $\{O[\delta(t)]^2\}$ are negligible and since $f(x_s)$ is 0, the system at steady state responds to small perturbations as $f(x) \approx \delta(t)f'(x_s)$. Since the left-hand side of the ODE equation

$$\frac{dx}{dt}$$

is equal to

$$\frac{d[x_s + \delta(t)]}{dt} = \frac{d\delta(t)}{dt}$$

the growth rate of perturbations is

$$\frac{d\delta(t)}{dt} = \delta(t)f'(x_s) \quad 6.10$$

Therefore, the perturbation will grow exponentially if $f'(x_s)$ is positive and will decay exponentially if $f'(x_s)$ is negative. Stability analysis of single-species systems is demonstrated in the gene expression example (See Section 3.1 for an example).

A bivariate system can be treated in a similar manner. For example, consider

$$\begin{aligned} \dot{x} &= f(x, y), & x &= x - \delta_x(t) \\ \dot{y} &= g(x, y), & y &= y - \delta_y(t) \end{aligned} \quad 6.11$$

where $\delta_x(t)$ and $\delta_y(t)$ denote a small disturbance from the steady-state solutions. Using the Taylor's expansion similar to the first-order system, we can approximate the growth rate of perturbations to be

$$\begin{pmatrix} \dot{\delta}_x \\ \dot{\delta}_y \end{pmatrix} = A \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix}, \text{ where } A = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix}_{(x_s, y_s)} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad 6.12$$

where A is the Jacobian matrix at a steady state. The exponents of the growth rate are determined by eigenvalues λ of the matrix A , given by the characteristic equation $\det(A - \lambda I)$, where I is the identity matrix. Defining $\tau = \text{trace}(A) = a + d$ and $\Delta = \det(A) = ad - bc$, the eigenvalues are

$$\lambda_1 = \frac{\tau + \sqrt{\tau^2 - 4\Delta}}{2}, \quad \lambda_2 = \frac{\tau - \sqrt{\tau^2 - 4\Delta}}{2} \quad 6.13$$

Since the real part of an eigenvalue determines the rate at which the perturbation grows, the real part of both eigenvalues must be negative for the steady-state solutions to be stable. General analysis for yet more complex biological systems can be found in [114].

Varying the parameter of interest can create or destroy steady-state solutions, and the properties of these solutions can change. At bifurcation points where the network behaviors undergo a qualitative change, a stable steady-state solution may become unstable or vice versa. Also, a stable steady-state solution may diverge to two or no steady states. We demonstrate practical use of bifurcation analysis in modeling a synthetic population control circuit (see Section 6.3.2 for an example). For an extensive treatment of bifurcation analysis, refer to [115].

6.3 Case Studies

To illustrate the basic concepts and techniques outlined above, we here provide examples of kinetic modeling and analysis using three simple biological systems: expression of a single gene, a phosphorylation-dephosphorylation cycle composed of enzymatic reactions, and a synthetic population control circuit.

6.3.1 Expression of a Single Gene

Although gene expression is a complicated process that involves a number of components, we use the simplistic view as shown in Figure 6.3(a). Key assumptions in this view are that transcription of mRNA is constitutive with rate k and that translation of protein depends on the concentration of mRNA. Although the choice of parameters depends on many factors such as the gene of interest and the internal and external environment of gene expression, the commonly accepted estimation of parameters is sufficient for our gene expression model. Based on simplification and estimated parameters, mathematical models are constructed using ODE, SDE, and

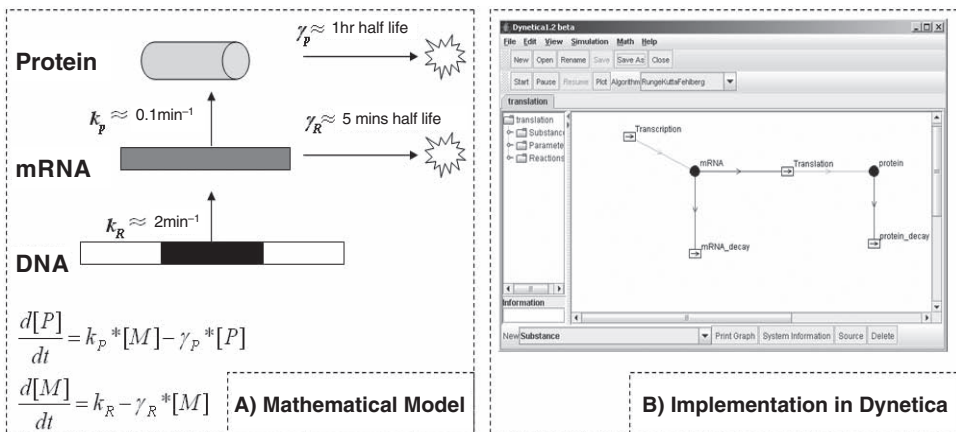


Figure 6.3 Modeling a single gene expression. A mathematical model is constructed based on our knowledge of the single gene expression and typical reaction parameters (a). While mRNA (M) is constitutively expressed with a rate constant of k_R , protein (P) is translated from mRNA with a rate constant of k_P . γ_P and γ_R are the degradation rate constants for the protein and the mRNA. This figure is adapted from [103]. (b) The model is implemented in a simulation and analysis software, *Dynetica* [82]. For direct comparison between stochastic and deterministic simulations, we chose to use molecular numbers as units for both proteins and mRNAs.

Table 6.4 Comparison between mathematical representation schemes for the gene expression.

ODE	SDE	Gillespie
<i>Reactions</i>	<i>Reactions</i>	<i>Reaction probability intensities^b</i>
$* \xrightarrow{k_R} \text{mRNA}$	$* \xrightarrow{k_R} \text{mRNA}$	$* \xrightarrow{c_R} \text{mRNA} \quad c_R$
$\text{mRNA} \xrightarrow{\gamma_R} *$	$\text{mRNA} \xrightarrow{\gamma_R} *$	$\text{mRNA} \xrightarrow{d_R} * \quad d_R N_R$
$* \xrightarrow{k_p[\text{mRNA}]} \text{protein}$	$* \xrightarrow{k_p[\text{mRNA}]} \text{protein}$	$* \xrightarrow{c_p N_R} \text{protein} \quad c_p N_R$
$\text{protein} \xrightarrow{\gamma_p} *$	$\text{protein} \xrightarrow{\gamma_p} *$	$\text{protein} \xrightarrow{d_p} * \quad d_p N_P$
<i>Ordinary differential equations</i>	<i>Stochastic differential equations^a</i>	<i>Gillespie Algorithm</i>
$\frac{d[\text{mRNA}]}{dt} = k_R - \gamma_R[\text{mRNA}]$	$\frac{d[\text{mRNA}]}{dt} = k_R - \gamma_R[\text{mRNA}] + \sqrt{k_R} \Gamma_1(t)$	• Calculate $a_0 = \sum_j^M a_j$
$\frac{d[\text{protein}]}{dt} = k_p *[\text{mRNA}] - \gamma_p[\text{protein}]$	$-\sqrt{\gamma_R[\text{mRNA}]} \Gamma_2(t)$	• Generate two random numbers r_1 and r_2
	$\frac{d[\text{protein}]}{dt} = k_p[\text{mRNA}] - \gamma_p[\text{protein}]$	• Compute $\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$
	$+\sqrt{k_p[\text{mRNA}]} \Gamma_3(t) - \sqrt{\gamma_p[\text{protein}]} \Gamma_4(t)$	• Compute μ that satisfies $\sum_i^\mu a_i \geq r_2 \times a_0 \geq \sum_j^{\mu-1} a_j$
		• Execute reaction μ and advance time t by τ

^a $\Gamma_i(t)$ ($i = 1, \dots, 4$) are temporally uncorrelated Gaussian noise.

^b c_R , d_R , c_p , and d_p are stochastic rate constants. $c_R = k_R N V$, where N is Avogadro's number and V is the cell volume. In this example, d_R , c_p , and d_p are the same as their corresponding conventional rate constants. N_R and N_P are the numbers of mRNA and protein molecules.

stochastic methods as shown in these models, implemented and simulated in a graphic-based simulator Dynetica [82] [Figure 6.3(b)]. Also see (<http://labs.genome.duke.edu/YouLab/software/dynetica/index.php>).

As shown by simulation results in Figure 6.4, the stochastic simulations generated dynamics similar overall to that from a deterministic simulation, but stochastic dynamics are noisy. The deterministic simulation also reveals that mRNA synthesis reaches steady state faster than protein production. Assuming a steady state for mRNA synthesis, we can carry out stability analysis of a steady state for gene expression. Equating the right-hand side of the ODE expression for mRNA in Table 6.4 to 0, we find the mRNA level at the steady state to be

$$\frac{K_R}{\gamma_R}$$

Then, the protein expression at the steady-state concentration of mRNA can be rewritten as

$$\frac{d[\text{protein}]}{dt} = \frac{k_p k_R}{\gamma_R} - \gamma_p[\text{protein}]$$

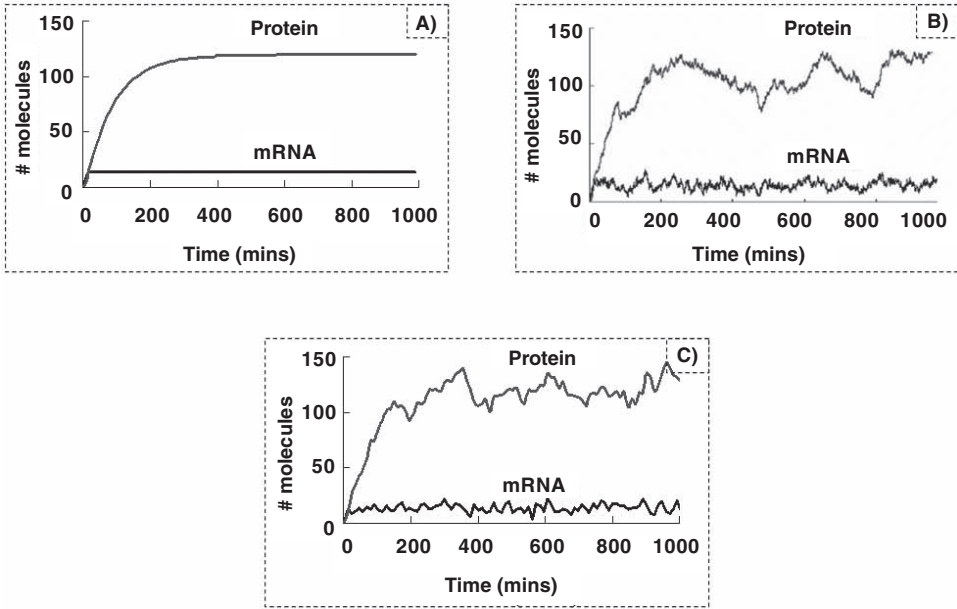


Figure 6.4 Simulation results of the model shown in Figure 6.3 by deterministic (a), SDE (b), and Gillespie (c) formulations.

When the decay rate ($\gamma_P[\text{protein}]$) matches the synthesis rate

$$\left(\frac{k_P k_R}{\gamma_R} \right)$$

the system is at a steady state. From (6.10), we can calculate the exponent for the growth rate of perturbation

$$\left(\frac{\partial f}{\partial P} \right)_{P=P_{ss}} = -\gamma_P < 0$$

where f is the right-hand side of the rate equation at the steady state, P is the protein level, and P_{ss} is the steady-state protein level. Since $f'(P_{ss})$ is negative, any perturbation around the steady state will decay at the rate of γ_P , indicating that the steady state is globally stable.

6.3.2 A Phosphorylation-Dephosphorylation Cycle

Increasing in complexity, we analyze transient and steady-state behaviors of an enzyme-mediated phosphorylation cycle, which has been shown to demonstrate ultrasensitivity when the enzymes operate outside the region of first-order kinetics [116]. To construct a mathematical model, we begin with the conventional enzyme catalysis scheme where a protein switches between its phosphorylated and dephosphorylated forms (Figure 6.5). Assuming the enzymatic reactions follow the Michaelis-Menten kinetics and the total protein concentration is constant, we develop two ODE equations, which are implemented and simulated in Dynetica. Since

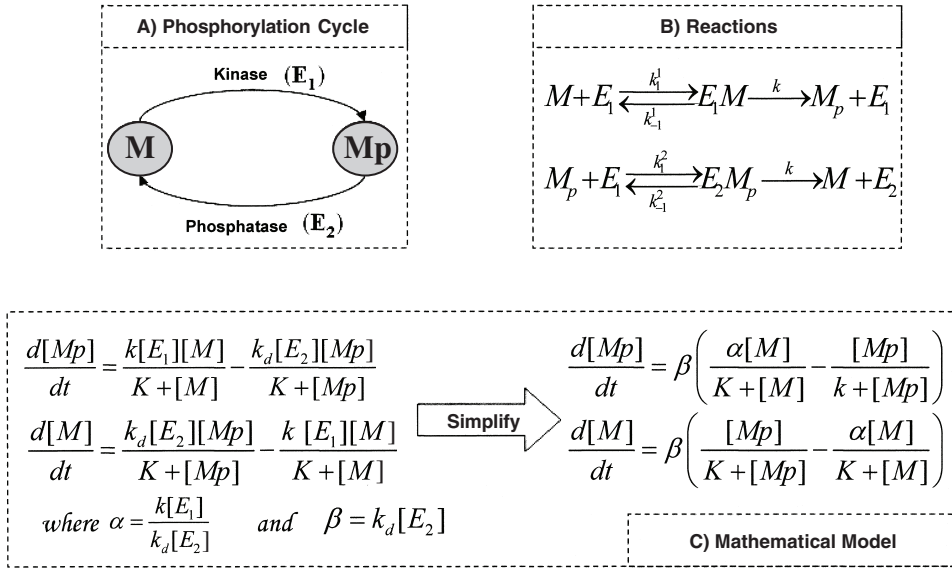


Figure 6.5 Modeling a phosphorylation-dephosphorylation cycle. An enzymatic modification cycle (a) of a protein between the dephosphorylated state M and the phosphorylated state M_p is mathematically modeled. k and k_d are rate constants for protein phosphorylation and dephosphorylation respectively. K is the Michaelis-Menten constant for the enzymatic modification cycle. Reaction schemes in (b) are converted to a set of ODEs (c) based on two assumptions: (1) Michaelis-Menten kinetics for the enzymatic reactions and (2) a constant total level of the protein.

the goal of modeling here is to identify general system behaviors of a phosphorylation cycle, we reason that a set of biologically feasible parameters in any phosphorylation cycles should be sufficient. Here, we choose parameters from the mitogen-activated protein kinase (MAPK) pathway [117], which has been extensively studied.

Assuming that the system starts with all protein in the unphosphorylated state, the protein will switch from the unphosphorylated state to the phosphorylated state over time, leading to a steady-state distribution of the protein in the two forms [Figure 6.6(a)]. This process is sensitive to α , a ratio between phosphorylation and dephosphorylation rates. When α is small, the amount of phosphorylated protein at the steady state is insignificant. However, more protein is converted as α becomes large. With very large α , the phosphorylation cycle becomes virtually irreversible, favoring the phosphorylated state.

The sensitivity analysis in Figure 6.6(b) shows the dependence of conversion on α . As the ratio of Michaelis-Menten constant to the total protein concentration, K , approaches 0, the dependence of conversion is ultrasensitive near α equal to 1. Then, the rate equation for the protein phosphorylation becomes

$$\frac{dM_p}{dt} = \beta(\alpha - 1)$$

a zero-order rate expression that does not depend on concentrations of reactants, products, or enzymes. This dynamics is thus called zero-order ultrasensitivity. When the Michaelis-Menten constants are comparable to the total protein concentration (large K), the rate expression is first order and the ultrasensitivity at $\alpha = 1$

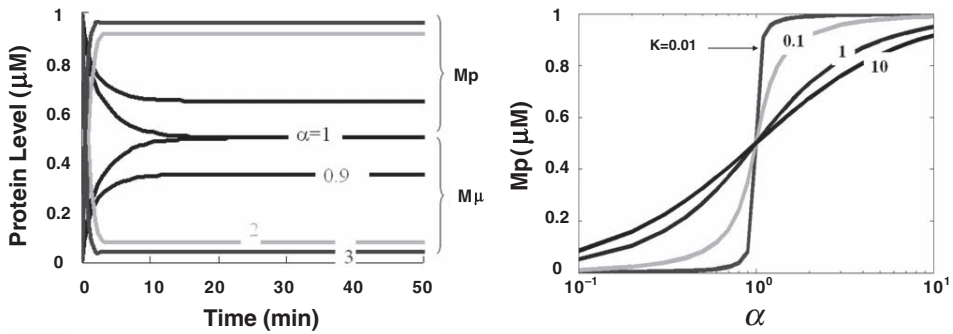


Figure 6.6 Simulation results for the model in Figure 6.5. Time-course results at varying α values show the dependence of conversion on the rate of phosphorylation and dephosphorylation (a). Protein conversion becomes ultrasensitive near $\alpha = 1$ for a sufficiently small Michaelis-Menten constant, while the sensitivity becomes weaker as K is increased (b).

becomes weaker. The time courses and sensitivity analysis in Figure 6.6 reveal two critical conditions to achieve ultrasensitivity: (1) α has to be near 1 and (2) the total protein concentration must be much greater than the Michaelis-Menten constants. That is, both kinase and phosphatase operate near saturation so that the overall reaction rate does not have a linear dependence on protein concentration.

We note that modeling can facilitate discovering design principles in biological systems. For example, ultrasensitivity is utilized in biological systems when sharp switching behavior is desired. A study of the mitogen-activated protein kinase (MAPK) pathway that combines both simulations and experiments has demonstrated ultrasensitivity. This work illustrates that the phosphorylation cycle mechanism under the two conditions is sufficient to generate a sharp switching behavior, whose Hill coefficient is estimated to be 5 [118]. At least two ways by which biological systems take advantage of ultrasensitivity can be speculated. In one scenario, a minor change in input will result in significant output when the system is operating near $\alpha = 1$. In the other scenario, a significant change in the input will have little impact on the output when α is much smaller or larger than 1. This may be useful in dealing with noisy signals, allowing the system to filter out noise [119].

6.3.3 A Synthetic Population Control Circuit

In addition to revealing dynamics of natural systems, modeling has become an indispensable tool for designing synthetic circuits [96, 120–125]. To illustrate this, we take as an example the synthetic population control circuit that we recently engineered [126, 127]. This circuit is based on a combination of two well-characterized modules: a quorum-sensing module and a killing module. Generally, we can develop an intuition about the circuit behavior, as the design is based on a combination of previously characterized modules. For example, the quorum-sensing module allows for cell-cell communication, where the cell density is broadcasted and detected by elements in the module. When the quorum-sensing module is coupled with a killing module, detection of high cell density by the quorum-sensing module activates killing of the cells. More specifically, the signal that diffuses across cell membranes to mediate communication is a small acyl-homoserine lactone (AHL)

molecule synthesized by the LuxI protein. At high cell density, the AHL accumulates inside the cells and in the extracellular medium. At sufficiently high concentrations, it activates the LuxR transcriptional regulator, which in turn activates expression of the killer gene (E) under the control of a LuxI promoter ($luxI$). Accumulation of the killer protein causes cell death. Based on this qualitative understanding of the programmed population control circuit, a set of ODE equations are formulated (Figure 6.7). The model is implemented, simulated, and analyzed in *XPP-AUT* [128].

To improve predictive power of the mathematical model, the model parameters are adjusted to reflect experimental results, which are variable depending on experimental conditions. In the population control circuit, for example, the degradation of AHL is shown to be facilitated by the medium pH condition [129]. A series of experiments with varying medium pH were performed to obtain circuit parameters, as shown in Table 6.5 [127]. In this study, accurate representation of the experimental behaviors required adjustment of the AHL degradation rate calculated from experimental results.

Once the parameters are determined from experimental results, the understanding of the network structure can give insights into the system dynamics. For example, the population control system has a negative feedback control on the cell density by the killer protein. Such a structure has the potential to generate complex

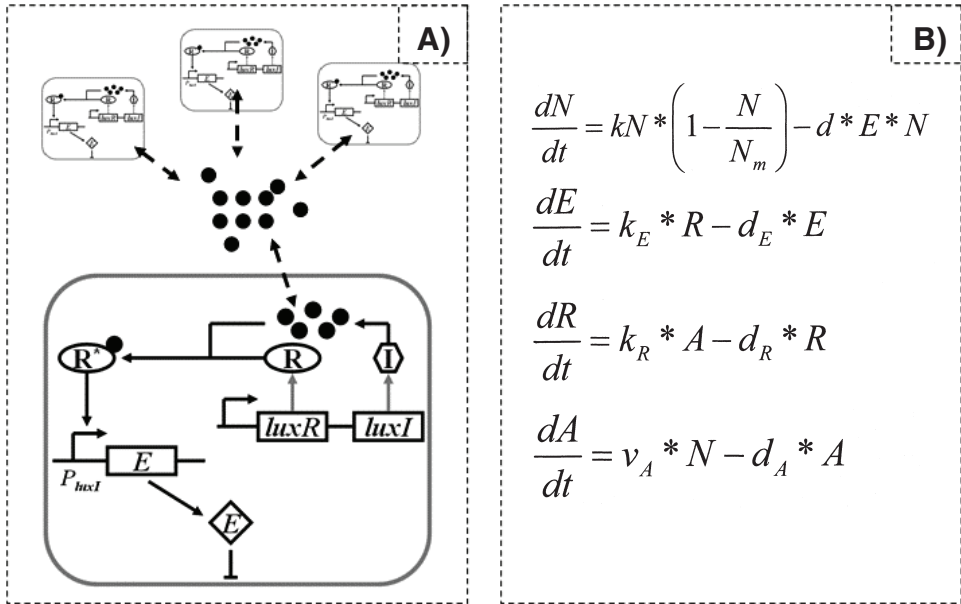


Figure 6.7 Modeling a synthetic population control circuit. (a) The cell density is broadcasted and detected by elements in the quorum-sensing module, and the killing of the cells is activated when high cell density is detected in the quorum-sensing module. The combination of the two modules allows for programmed population control. (b) An ODE-based kinetic model. Viable cell density N follows logistic growth with a specific growth rate k and a carrying capacity N_m . AHL (A), synthesized from the viable cells with a rate constant v_A , activates LuxR (R) with a rate constant k_R . Here d is the killing rate constant. k_E is the synthesis rate constant of E catalyzed by active LuxR and d_E , d_R , and d_A are the degradation rate constants for E , R , and A respectively. More details on the assumptions and the parameters can be found in [126, 127].

Table 6.5 Effects of pH on circuit parameters (adapted from [127]).

Medium pH	k (h^{-1})	$N_m/10^9$ (CFU ml^{-1})	$N_s/10^7$ (CFU ml^{-1})	d_A (h^{-1})
6.2	0.885	1.25 ± 0.06	4.86 ± 0.02	0.274
6.6	0.928	1.17 ± 0.05	5.59 ± 0.03	0.304
7.0	0.970	1.24 ± 0.10	11.7 ± 0.6	0.639
7.4	0.897	1.16 ± 0.10	13.1 ± 0.6	0.791
7.8	0.936	1.20 ± 0.07	19.5 ± 1.3	1.19

dynamics, including oscillations. For certain biologically feasible parameters, our analysis shows that the model can indeed generate sustained oscillations over time. This prediction is consistent with experimental observations [126]. Further system stability analysis indicates that for $N \ll N_m$, there are two steady-state solutions. While the trivial steady state is always unstable, the nontrivial steady state is stable if degradation rates of LuxR, the killer protein, the AHL signal, and the microchemostat dilution rates are sufficiently large. However, decreases in these parameters destabilize the nontrivial steady state, leading to oscillations. This trend is captured in Figure 6.8. For each of these parameters, bifurcation analysis is carried out using *XPP-AUT* [Figure 6.9(a)]. In Figure 6.9b oscillations are observed for d_A less than 0.35, and the amplitude of the oscillations is the difference between the top and the bottom curves. High values of d_A (>0.35) stabilize the system, and the magnitude of oscillations decreases until damped oscillations occur [Figure 6.9(c)]. Further increases in d_A lead to stronger dampening of the oscillations that eventually eliminate oscillations [Figure 6.9(d)]. Similar stability analysis is carried out for the other parameters, and similar behaviors of the nontrivial steady-state solution are observed (Figure 6.10).

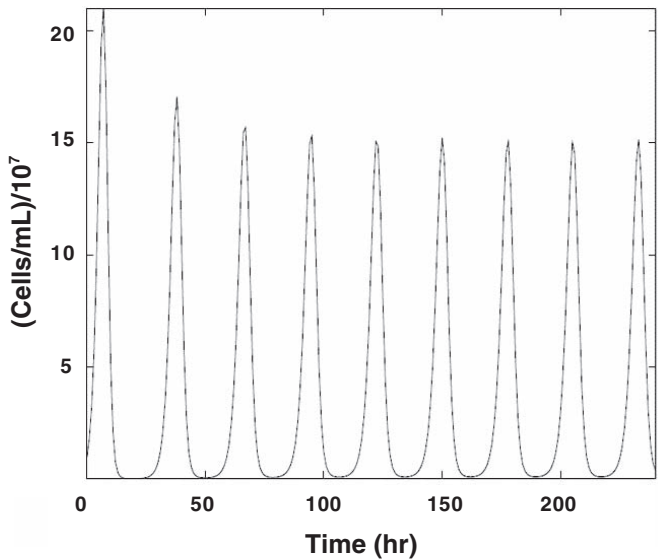


Figure 6.8 Oscillation in the cell density over time for appropriate parameter values.

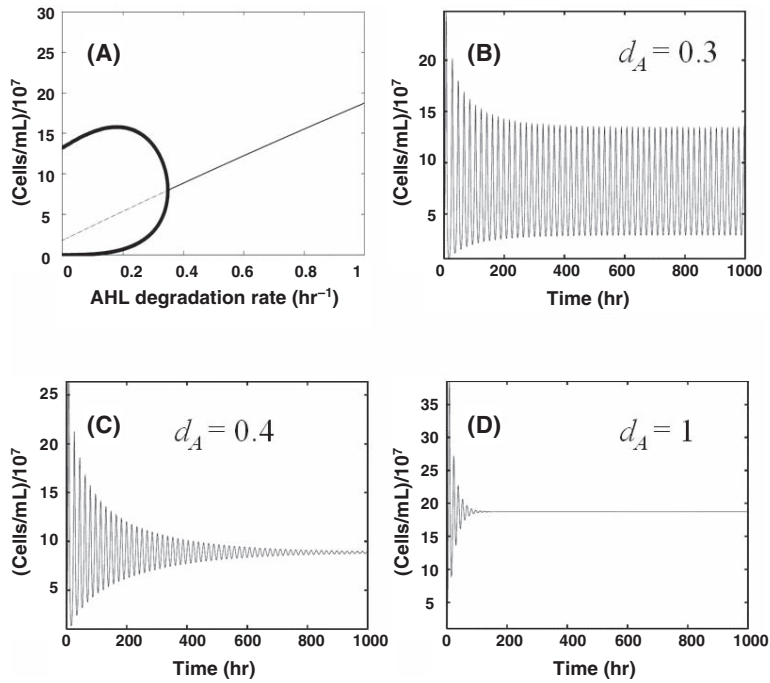


Figure 6.9 Bifurcation analysis. (a) Qualitative changes are observed in the dynamics as the AHL degradation rate constant (d_A) is varied. For sufficiently large d_A the steady-state solutions are stable as represented by the thin line. As d_A is decreased, the steady-state solutions become unstable (dashed line) and exhibit oscillations. The top and bottom branches of oscillations are indicated by the upper and lower curves respectively. For example, (b) oscillation in cell density is observed when d_A is sufficiently small (≤ 0.35). (c) The population undergoes damped oscillation in cell density for increased d_A . (d) Further increase in d_A stabilizes cell density.

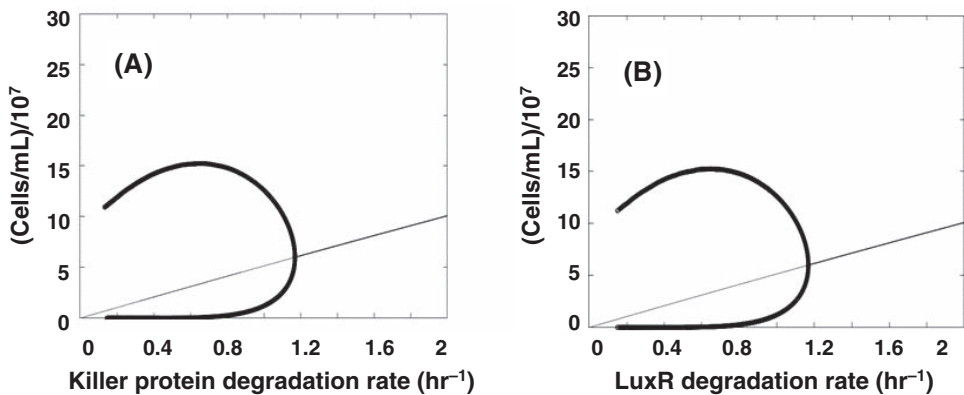


Figure 6.10 Further bifurcation analysis with rates for killer protein degradation (a) and LuxR degradation (b) is performed. Oscillations at sufficiently smaller rates diminish as the rates increase.

6.4 Conclusion

We have used relatively simple, well-characterized systems to illustrate construction and analysis of kinetic models. In these simple examples, we have demonstrated the significance of kinetic modeling not only for improved understanding of biological systems but also for improved predictions of cellular response to perturbations. We note that mathematical modeling is not limited to simple systems, but has also been used in more complex systems. Successful application of modeling has been demonstrated by numerous studies. The increase in the complexity of modeled systems suggests wider applicability of mathematical modeling. Integrated understanding of complex systems, whose dynamics cannot be conceptualized by intuition alone, can be achieved in a quantitative manner. Also, improved predictive power is particularly promising in the development of therapeutics, where system-level understanding is essential to minimize side effects and to precisely predict drug effects. Finally, modeling of cellular networks has become an integral part of the nascent field of synthetic biology. The combination of design, modeling, experimental implementation, and characterization of synthetic circuits or modules can provide substantial insights into the design principles of more complex natural biological systems and assist in the creation of artificial systems for practical applications.

References

- [1] Alemany, R., C. Balague, and D. T. Curiel, "Replicative adenoviruses for cancer therapy," *Nature Biotechnol.*, Vol. 18, No. 7, 2000, pp. 723–727.
- [2] Bischoff, J. R., et al., "An adenovirus mutant that replicates selectively in p53-deficient human tumor cells," *Science*, 1996, Vol. 274, No. 5286, pp. 373–376.
- [3] Coffey M. C., et al., "Reovirus therapy of tumors with activated Ras pathway," *Science*, Vol. 282, No. 5392, 1998, pp. 1332–1334.
- [4] Guillemard, V., and H. U. Saragovi, "Novel approaches for targeted cancer therapy," *Curr. Cancer Drug Targets*, Vol. 4, No. 4, 2004, pp. 313–326.
- [5] Jakubczak J. L., et al., "An oncolytic adenovirus selective for retinoblastoma tumor suppressor protein pathway-defective tumors: dependence on E1A, the E2F-1 promoter, and viral replication for selectivity and efficacy," *Cancer Res.*, Vol. 63, No. 7, 2003, pp. 1490–1499.
- [6] Nevins, J. R., "The Rb/E2F pathway and cancer," *Hum. Mol. Genet.*, Vol. 10, No. 7, 2001, pp. 699–703.
- [7] Rogulski, K. R., et al., "Double suicide gene therapy augments the antitumor activity of a replication-competent lytic adenovirus through enhanced cytotoxicity and radiosensitization," *Hum. Gene Therapy*, Vol. 11, No. 1, 2000, pp. 67–76.
- [8] Johnson, L., et al., "Selectively replicating adenoviruses targeting deregulated E2F activity are potent, systemic antitumor agents," *Cancer Cell*, Vol. 1, No. 4, 2002, pp. 325–337.
- [9] Khuri, F. R., et al., "A controlled trial of intratumoral ONYX-015, a selectively-replicating adenovirus, in combination with cisplatin and 5-fluorouracil in patients with recurrent head and neck cancer," *Nat. Med.*, Vol. 6, No. 8, 2000, pp. 879–885.
- [10] Hasty, J., et al., "Computational studies of gene regulatory networks: in numero molecular biology," *Nat. Rev. Genet.*, Vol. 2, No. 4, 2001, pp. 268–279.
- [11] Ideker, T., L. Winslow, and A. Lauffenburger, "Bioengineering and Systems Biology," *Annals Biomed. Eng.*, Vol. 34, No. 2, 2006, p. 257.

- [12] Neves, S. R., "Modeling of signaling networks," *BioEssays*, Vol. 24, No. 12, 2002, p. 1110.
- [13] Weston, A. D., "Systems biology, proteomics, and the future of health care: Toward predictive, preventative, and personalized medicine," *J. Proteome Res.*, Vol. 3, No. 2, 2004, p. 179.
- [14] Kitano, H., "Computational systems biology," *Nature*, Vol. 420, No. 6912, 2002, p. 206.
- [15] Rao, C. V., A. P. Arkin, "Control motifs for intracellular regulatory networks," *Annu. Rev. Biomed. Eng.*, Vol. 3, 2001, pp. 391–419.
- [16] Endy, D., and R. Brent, "Modelling cellular behaviour," *Nature*, Vol. 409, No. 6818, 2001, pp. 391–395.
- [17] Kholodenko, B. N., "Cell-signalling dynamics in time and space," *Nat. Rev. Mol. Cell Biol.*, Vol. 7, No. 3, 2006, pp. 165–176.
- [18] Alves, R., F. Antunes, and A. Salvador, "Tools for kinetic modeling of biochemical networks," *Nat. Biotech.*, Vol. 24, No. 6, 2006, p. 667.
- [19] You, L., "Toward computational systems biology," *Cell Biochem. Biophys.*, 40, No. 2, 2004, pp. 167–184.
- [20] Sasagawa, S., et al., "Prediction and validation of the distinct dynamics of transient and sustained ERK activation," *Nat. Cell Biol.*, Vol. 7, No. 4, 2005, pp. 365–373.
- [21] Schoeberl, B., et al., "Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors," *Nat. Biotechnol.*, Vol. 20, No. 4, 2002, pp. 370–375.
- [22] Bhalla, U. S., P. T. Ram, and R. Iyengar, "MAP kinase phosphatase as a locus of flexibility in a mitogen-activated protein kinase signaling network," *Science*, Vol. 297, No. 5583, 2002, pp. 1018–1023.
- [23] Asthagiri, A. R., and D. A. Lauffenburger, "A computational study of feedback effects on signal dynamics in a mitogen-activated protein kinase (MAPK) pathway model," *Biotechnol. Progress*, Vol. 17, No. 2, 2001, pp. 227–239.
- [24] Bhalla, U. S., and R. Iyengar, "Emergent properties of networks of biological signaling pathways," *Science*, Vol. 283, No. 5400, 1999, pp. 381–387.
- [25] Taniguchi, C. M., B. Emanuelli, and C. R. Kahn, "Critical nodes in signalling pathways: insights into insulin action," *Nat. Rev. Mol. Cell Biol.*, Vol. 7, No. 2, 2006, pp. 85–96.
- [26] Somogyi, R., and L. D. Greller, "The dynamics of molecular networks: applications to therapeutic discovery," *Drug Discovery Today*, Vol. 6, No. 24, 2001, pp. 1267–1277.
- [27] Jackson, T. L., and H. M. Byrne, "A mathematical model to study the effects of drug resistance and vasculature on the response of solid tumors to chemotherapy," *Math. Biosci.*, Vol. 164, No. 1, 2000, pp. 17–38.
- [28] Butcher, E. C., E. L. Berg, and E. J. Kunkel, "Systems biology in drug discovery," *Nat. Biotechnol.*, Vol. 22, No. 10, 2004, pp. 1253–1259.
- [29] Oda, K., et al., "A comprehensive pathway map of epidermal growth factor receptor signaling," *Mol. Syst. Biol.*, Vol. 1, No. 1, 2005, msb4100014–E4100011.
- [30] Gardner, T. S., et al., "Inferring genetic networks and identifying compound mode of action via expression profiling," *Science*, Vol. 301, No. 5629, 2003, pp. 102–105.
- [31] Butte, A. J., and I. S. Kohane, "Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements," *Pacific Symp. Biocomputing*, 2000, pp. 418–429.
- [32] Friedman, N., et al., "Using Bayesian networks to analyze expression data," *J. Comput. Biol.*, Vol. 7, No. 3-4, 2000, pp. 601–620.
- [33] Moles, C. G., P. Mendes, and J. R. Banga, "Parameter estimation in biochemical pathways: A comparison of global optimization methods," *Genome Res.*, Vol. 13, No. 11, 2003, pp. 2467–2474.

- [34] Arkin, A., P. Shen, and J. Ross, "A Test Case of Correlation Metric Construction of a Reaction Pathway from Measurements," *Science*, Vol. 277, No. 5330, 1997, pp. 1275–1279.
- [35] You, L., and J. Yin, "Patterns of regulation from mRNA and protein time series," *Metabolic Eng.*, Vol. 2, No. 3, 2000, pp. 210–217.
- [36] Ideker, T., et al., "Testing for differentially-expressed genes by maximum-likelihood analysis of microarray data," *J. Compu. Biol.*, Vol. 7, No. 6, 2000, pp. 805–817.
- [37] Ronen, M., et al., "Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics," *Proc. Natl. Acad. Sci. USA*, Vol. 99, No. 16, 2002, pp. 10555–10560.
- [38] Guido, N. J., et al., "A bottom-up approach to gene regulation," *Nature*, Vol. 439, No. 7078, 2006, pp. 856–860.
- [39] Austin, D. W., et al., "Gene network shaping of inherent noise spectra," *Nature*, Vol. 439, No. 7076, 2006, pp. 608–611.
- [40] Elowitz, M. B., et al., "Stochastic gene expression in a single cell," *Science*, Vol. 297, No. 5584, 2002, pp. 1183–1186.
- [41] Pedraza, J. M., and A. van Oudenaarden, "Noise propagation in gene networks," *Science*, Vol. 307, No. 5717, 2005, pp. 1965–1969.
- [42] Becskei, A., B. B. Kaufmann, and A. van Oudenaarden, "Contributions of low molecule number and chromosomal positioning to stochastic gene expression," *Nat. Genet.*, Vol. 37, No. 9, 2005, pp. 937–944.
- [43] Rosenfeld, N., et al., "Gene regulation at the single-cell level," *Science*, Vol. 307, No. 5717, 2005, pp. 1962–1965.
- [44] Keseler, I. M., et al., "A comprehensive database resource for *Escherichia coli*," *Nucleic Acids Res.*, Vol. 33, Database issue, 2005, pp. D334–D337.
- [45] Kanehisa, M., et al., "From genomics to chemical genomics: new developments in KEGG," *Nucleic Acids Res.*, Vol. 34, Database issue, 2006, pp. D354–D357.
- [46] Overbeek, R., et al., "The ERGO genome analysis and discovery system," *Nucleic Acids Res.*, Vol. 31, No. 1, 2003, pp. 164–171.
- [47] Lemer, C., et al., "The aMAZE LightBench: a web interface to a relational database of cellular processes," *Nucleic Acids Res.*, Vol. 32, Database issue, 2004, pp. D443–D448.
- [48] Gasteiger, E., et al., "ExPASy: The proteomics server for in-depth protein knowledge and analysis," *Nucleic Acids Res.*, 31, No. 13, 2003, pp. 3784–3788.
- [49] Wolf, D. M., and A. P. Arkin, "Motifs, modules and games in bacteria," *Curr. Opinion Microbiol.*, Vol. 6, No. 2, 2003, pp. 125–134.
- [50] Hartwell, L. H., et al., "From molecular to modular cell biology," *Nature*, Vol. 402, Suppl. 6761, 1999, pp. C47–C52.
- [51] Romond, P.-C., et al., "Alternating Oscillations and Chaos in a Model of Two Coupled Biochemical Oscillators Driving Successive Phases of the Cell Cycle," *Annals NY Acad. Sci.*, Vol. 879, No. 1, 1999, pp. 180–193.
- [52] Tyson, J. J., K. C. Chen, and B. Novak, "Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell," *Curr. Opinion Cell Biol.*, Vol. 15, No. 2, 2003, pp. 221–231.
- [53] Batchelor, E., T. J. Silhavy, and M. Goulian, "Continuous control in bacterial regulatory circuits," *J. Bacteriol.*, Vol. 186, No. 22, 2004, pp. 7618–7625.
- [54] Becskei, A., and L. Serrano, "Engineering stability in gene networks by autoregulation," *Nature*, Vol. 405, No. 6786, 2000, pp. 590–593.
- [55] Rosenfeld, N., M. B. Elowitz, and U. Alon, "Negative autoregulation speeds the response times of transcription networks," *J. Mol. Biol.*, Vol. 323, No. 5, 2002, pp. 785–793.
- [56] Becskei, A., B. Seraphin, and L. Serrano, "Positive feedback in eukaryotic gene networks: cell differentiation by graded to binary response conversion," *EMBO J.*, Vol. 20, No. 10, 2001, pp. 2528–2535.

- [57] Kramer, B. P., and M. Fussenegger, "Hysteresis in a synthetic mammalian gene network," *Proc. Natl. Acad. Sci. USA*, Vol. 102, No. 27, 2005, pp. 9517–9522.
- [58] Thron, C. D., "Bistable biochemical switching and the control of the events of the cell cycle," *Oncogene*, Vol. 15, No. 3, 1997, pp. 317–325.
- [59] Yao, G., et al., In preparation.
- [60] Acar, M., A. Becskei, and A. van Oudenaarden, "Enhancement of cellular memory by reducing stochastic transitions," *Nature*, Vol. 435, No. 7039, 2005, pp. 228–232.
- [61] Gardner, T. S., C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in *Escherichia coli*," *Nature*, Vol. 403, No. 6767, 2000, pp. 339–342.
- [62] Atkinson, M. R., et al., "Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in *Escherichia coli*," *Cell*, Vol. 113, No. 5, 2003, pp. 597–607.
- [63] Elowitz, M. B., and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, Vol. 403, No. 6767, 2000, pp. 335–338.
- [64] Fung, E., et al., "A synthetic gene-metabolic oscillator," *Nature*, 2005, Vol. 435, No. 7038, pp. 118–122.
- [65] Guantes, R., and J. F. Poyatos, "Dynamical principles of two-component genetic oscillators," *PLoS Compu. Biol.*, Vol. 2, No. 3, 2006, p. e30.
- [66] Milo, R., et al., "Network motifs: simple building blocks of complex networks," *Science*, Vol. 298, No. 5594, 2002, pp. 824–827.
- [67] Shen-Orr, S. S., et al., "Network motifs in the transcriptional regulation network of *Escherichia coli*," *Nat. Genet.*, Vol. 31, No. 1, 2002, pp. 64–68.
- [68] Stoleru, D., et al., "A resetting signal between *Drosophila* pacemakers synchronizes morning and evening activity," *Nature*, Vol. 438, No. 7065, 2005, pp. 238.
- [69] Levine, J. D., et al., "Signal analysis of behavioral and molecular cycles," *BMC Neurosci.*, Vol. 3, 2002, p. 1.
- [70] Meinhardt, H., "Pattern formation in biology: a comparison of models and experiments," *Reps. Progr. Physics*, Vol. 55, No. 6, 1992, p. 797.
- [71] Xiong, W., and J. E. Ferrell, Jr., "A positive-feedback-based bistable 'memory module' that governs a cell fate decision," *Nature*, Vol. 426, No. 6965, 2003, pp. 460–465.
- [72] Tyson, J. J., and B. Novak, "Regulation of the eukaryotic cell cycle: molecular antagonism, hysteresis, and irreversible transitions," *J. Theoret. Biol.*, Vol. 210, No. 2, 2001, pp. 249–263.
- [73] Tyson, J. J., et al., "Checkpoints in the cell cycle from a modeler's perspective," *Progr. Cell Cycle Res.*, Vol. 1, 1995, pp. 1–8.
- [74] Pomerening, J. R., E. D. Sontag, and J. E. Ferrell, Jr., "Building a cell cycle oscillator: hysteresis and bistability in the activation of Cdc2," *Nat. Cell Biol.*, Vol. 5, No. 4, 2003, pp. 346–351.
- [75] Bintu, L., et al., "Transcriptional regulation by the numbers: applications," *Curr. Opinion Genet. Devel.*, 15, No. 2, 2005, pp. 125–135.
- [76] Bintu, L., et al., "Transcriptional regulation by the numbers: models," *Curr. Opinion Genet. Devel.*, 15, No. 2, 2005, pp. 116–124.
- [77] Mathews, J. H., and K. D. Fink, *Numerical Methods Using MATLAB*, 4th ed., Upper Saddle River, NJ: Pearson, 2004.
- [78] Atkinson, K. E., *An Introduction to Numerical Analysis*, 2nd ed., New York: Wiley, 1989.
- [79] Quarteroni, A., R. Sacco, and F. Saleri, *Numerical Mathematics*, New York: Springer, 2000.
- [80] Epperson, J. F., *An Introduction to Numerical Methods and Analysis*, New York: John Wiley, 2002.
- [81] Slepchenko, B. M., et al., "Quantitative cell biology with the Virtual Cell," *Trends Cell Biol.*, Vol. 13, No. 11, 2003, pp. 570–576.

- [82] You, L., A. Hoonlor, and J. Yin, "Modeling biological systems using Dynetica—a simulator of dynamic networks," *Bioinformatics*, Vol. 19, No. 3, 2003, pp. 435–436.
- [83] Ramsey, S., D. Orrell, and H. Bolouri, "Dizzy: stochastic simulation of large-scale genetic regulatory networks," *J. Bioinformatics Compu. Biol.*, Vol. 3, No. 2, 2005, pp. 415–436.
- [84] Dhar, P., et al., "Cellware—a multi-algorithmic software for computational systems biology," *Bioinformatics*, Vol. 20, No. 8, 2004, pp. 1319–1321.
- [85] Hucka, M., et al., "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models," *Bioinformatics*, Vol. 19, No. 4, 2003, pp. 524–531.
- [86] Mendes, P., "Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3," *Trends Biochem. Sci.*, Vol. 22, No. 9, 1997, pp. 361–363.
- [87] McAdams, H. H., and A. Arkin, "Stochastic mechanisms in gene expression," *Proc. Natl. Acad. Sci. USA*, Vol. 94, No. 3, 1997, pp. 814–819.
- [88] Arkin, A., J. Ross, and H. H. McAdams, "Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected *Escherichia coli* cells," *Genetics*, Vol. 149, No. 4, 1998, pp. 1633–1648.
- [89] Kaern, M., et al., "Stochasticity in gene expression: from theories to phenotypes," *Nat. Rev. Genet.*, Vol. 6, No. 6, 2005, pp. 451–464.
- [90] Hooshangi, S., S. Thiberge, and R. Weiss, "Ultrasensitivity and noise propagation in a synthetic transcriptional cascade," *Proc. Natl. Acad. Sci. USA*, Vol. 102, No. 10, 2005, pp. 3581–3586.
- [91] Raser, J. M., and E. K., "Control of stochasticity in eukaryotic gene expression," *Science*, Vol. 304, No. 5678, 2004, pp. 1811–1814.
- [92] Bar-Even, A., et al., "Noise in protein expression scales with natural protein abundance," *Nat. Genet.*, Vol. 38, No. 6, 2006, pp. 636–643.
- [93] Rao, C. V., D. M. Wolf, and A. P. Arkin, "Control, exploitation and tolerance of intracellular noise" [erratum appears in *Nature*, Vol. 421, No. 6919, 9 Jan. 2003, p. 190.], *Nature*, Vol. 420, No. 6912, 2002, pp. 231–237.
- [94] Savageau, M. A., "Comparison of classical and autogenous systems of regulation in inducible operons," *Nature*, Vol. 252, No. 5484, 1974, pp. 546–549.
- [95] Weinberger, L. S., et al., "Stochastic gene expression in a lentiviral positive-feedback loop: HIV-1 Tat fluctuations drive phenotypic diversity," *Cell*, Vol. 122, No. 2, 2005, pp. 169–182.
- [96] Suel, G. M., et al., "An excitable gene regulatory circuit induces transient cellular differentiation," *Nature*, Vol. 440, No. 7083, 2006, pp. 545–550.
- [97] Gillespie, D. T., "A rigorous derivation of the chemical master equation," *Physica A: Statistical and Theoretical Physics*, Vol. 188, No. 1-3, 1992, pp. 404–425.
- [98] Gillespie, D. T., "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, Vol. 81, No. 25, 1977, pp. 2340.
- [99] Gibson, M. A., and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many reactions," *J. Chem. Phys.*, Vol. 104, 2000, pp. 1876–1889.
- [100] Cao, Y., D. T. Gillespie, and L. R. Petzold, "Efficient step size selection for the tau-leaping simulation method," *J. Chem. Phys.*, Vol. 124, No. 4, 2006, p. 044109.
- [101] Haseltine, E. L., and J. B. Rawlings, "Approximate simulation of coupled fast and slow reactions stochastic chemical kinetics," *J. Chem. Phys.*, Vol. 117, No. 15, 2002, pp. 6959–6969.
- [102] Gillespie, D. T., "The chemical Langevin equation," *J. Chem. Phys.*, Vol. 113, No. 1, 2000, pp. 297–306.
- [103] Ozbudak, E. M., et al., "Regulation of noise in the expression of a single gene," *Nat. Genet.*, Vol. 31, No. 1, 2002, pp. 69–73.

- [104] Fell, D. A., "Metabolic control analysis: a survey of its theoretical and experimental development," *Biochem. J.*, Vol. 286, Pt. 2, 1992, pp. 313–330.
- [105] Varma, A., M. Morbidelli, and H. Wu, *Parametric Sensitivity in Chemical Systems*, Cambridge, UK/New York, NY: Cambridge Univ. Press, 1999.
- [106] Morohashi, M., et al., "Robustness as a measure of plausibility in models of biochemical networks," *J. Theoret. Biol.*, Vol. 216, No. 1, 2002, pp. 19–30.
- [107] Barkai, N., and S. Leibler, "Robustness in simple biochemical networks," *Nature*, Vol. 387, No. 6636, 1997, pp. 913–917.
- [108] You, L., and J. Yin, "Dependence of epistasis on environment and mutation severity as revealed by in silico mutagenesis of phage $\tau 7$," *Genetics*, Vol. 160, No. 4, 2002, pp. 1273–1281.
- [109] You, L., and J. Yin, "Evolutionary design on a budget: robustness and optimality of bacteriophage T7," *IEE Proc. Systems Biol.*, Vol. 153, No. 2, 2006, pp. 46–52.
- [110] Alon, U., et al., "Robustness in bacterial chemotaxis," *Nature*, Vol. 397, No. 6715, 1999, pp. 168–171.
- [111] Freeman, M., "Feedback control of intercellular signalling in development," *Nature*, Vol. 408, No. 6810, 2000, p. 313.
- [112] Csete, M. E., and J. C. Doyle, "Reverse Engineering of Biological Complexity," *Science*, Vol. 295, No. 5560, 2002, pp. 1664–1669.
- [113] Carlson, J. M., and J. Doyle, "Complexity and robustness," *Proc. Natl. Acad. Sci. USA*, Vol. 99, No. 90001, 2002, pp. 2538–2545.
- [114] Murray, J. D., *Mathematical Biology*, 2nd. corr. ed., Berlin/New York: Springer-Verlag, 1993.
- [115] Strogatz, S. H., *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, Reading, MA: Addison-Wesley Pub., 1994.
- [116] Goldbeter, A., and D. E. Koshland, Jr., "An amplified sensitivity arising from covalent modification in biological systems," *Proc. Natl. Acad. Sci. USA*, Vol. 78, No. 11, 1981, pp. 6840–6844.
- [117] Kholodenko, B. N., "Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades," *Eur. J. Biochem.*, Vol. 267, No. 6, 2000, pp. 1583–1588.
- [118] Huang, C. Y., and J. E. Ferrell, Jr., "Ultrasensitivity in the mitogen-activated protein kinase cascade," *Proc. Natl. Acad. Sci. USA*, Vol. 93, No. 19, 1996, pp. 10078–10083.
- [119] Ferrell, J. E., Jr., "Tripping the switch fantastic: how a protein kinase cascade can convert graded inputs into switch-like outputs," *Trends Biochem. Sci.*, Vol. 21, No. 12, 1996, p. 460.
- [120] Tu, D., et al., "Engineering Gene Circuits: Foundations and applications," *Nanotechnol. Biotechnol. Med.*, 2006.
- [121] Church, G. M., "From systems biology to synthetic biology," *Mol. Syst. Biol.*, Vol. 1, No. 1, 2005, pp. msb4100007–E4100001.
- [122] Hasty, J., D. McMillen, and J. J. Collins, "Engineered gene circuits," *Nature*, Vol. 420, No. 6912, 2002, pp. 224–230.
- [123] Andrianantoandro, E., et al., "Synthetic biology: new engineering rules for an emerging discipline," *Mol. Syst. Biol.*, Vol. 2, 2006, p. E1.
- [124] Colman-Lerner, A., et al., "Regulated cell-to-cell variation in a cell-fate decision system," *Nature*, Vol. 437, No. 7059, 2005, pp. 699–706.
- [125] Endy, D., "Foundations for engineering biology," *Nature*, Vol. 438, No. 7067, 2005, p. 449.
- [126] Balagadde, F. K., et al., "Long-term monitoring of bacteria undergoing programmed population control in a microchemostat," *Science*, Vol. 309, No. 5731, 2005, pp. 137–140.

- [127] You, L., et al., “Programmed population control by cell-cell communication and regulated killing,” *Nature*, Vol. 428, No. 6985, No., 2004, pp. 868–871.
- [128] Doedel, E. J., “AUTO: A program for the automatic bifurcation analysis of autonomous systems,” *Dynamics*, Vol. 38, No. 9, 1983, p. 1493.
- [129] Schaefer, A. L., et al., “Detection, purification, and structural elucidation of the acylhomoserine lactone inducer of *Vibrio fischeri* luminescence and other related molecules,” *Methods Enzymol.*, Vol. 305, 2000, pp. 288–301.

PART IV

Analysis: Probabilistic Data Networks and Communications

Topological Analysis of Biomolecular Networks

Vinayak Muralidhar, Gabor Szabo, and Gil Alterovitz

7.1 Cellular Networks

Topology, as used in this chapter, is the study of cellular networks using a graph-theoretical perspective. Cellular networks are viewed in terms of nodes, which can represent genes, proteins, or metabolic compounds, and edges, which are interactions among the nodes. The study of the topology of cellular networks is important because it elucidates characteristics about organisms that would otherwise go undetected and it has a wide array of other potential applications in computational biology. This chapter deals with the use of network topology in extracting information from cellular networks as well as studying essential genes and their computational detection. The topological study in this chapter refers only to computational biology. Networks are studied without any major participation of *in vivo* or *in vitro* analysis, except to corroborate *in silico* results.

The majority of this chapter's focus is on the organism *Escherichia coli*. *E. coli* is a common bacterium, often known for its pathogenic properties. *E. coli* has, however, been studied very well since the 1970s. The EcoCyc database is a compilation of much of the data collected and is the primary basis for the discussion of *E. coli* in this chapter in regard to essential gene prediction [1]. *E. coli* is considered the model organism for prokaryotes [2], and the metabolism and genetics of *E. coli* are well documented [3]. However, conclusions pertaining to *E. coli* are not limited to only the bacteria species. By the biological principle of conservation, important features are conserved through evolution because they are advantageous to express [4, 5]. Thus, it is possible to generalize the discussion of *E. coli* here to all prokaryotes and possibly even to all organisms [6].

This chapter also discusses the most common model organism for eukaryotes, *Saccharomyces cerevisiae*, a species of budding yeast. *S. cerevisiae* is common yeast, also known as baker's yeast. Many researchers choose to study *S. cerevisiae* because it is already well studied, is easy to culture, and is biochemically amenable [7, 8]. Furthermore, its initial role as a eukaryotic model organism was due to its established presence in various industries, such as in the bakery and brewery industries [8].

Although computational biology studies organisms from a computer science standpoint, the building blocks of the field as well as most of its impacts belong to biology. This chapter focuses on the role of the proteome in the cell. The *proteome* is the entire set of an organism's proteins and the various interactions among and involving these proteins [9]. However, before the age of proteomics was the age of genomics. From the field of genomics, many other fields developed, such as proteomics, computational evolutionary biology, and protein structure prediction. Genomics studies the entire set of an organism's genes. A *gene* is the unit of heredity in all living organisms. Genes are encoded in segments of DNA nucleotides. Each gene codes for either a polypeptide or an RNA molecule, such as rRNA, tRNA, or mRNA. Polypeptides are polymers of amino acids that fold and form proteins. Proteins are macromolecules that carry out almost all of the functions in a cell. Everything from cellular defense and metabolizing inputs to replicating DNA and performing cytokinesis is performed with the help of proteins in a cell. These proteins participate in three major networks in cells: genetic regulation networks, protein-protein networks, and metabolic regulation networks.

After introducing the various cellular networks commonly studied by researchers in computational biology and after establishing some general terms, this chapter focuses on two areas of cellular network topology. First, this chapter discusses properties of each of the cellular networks individually and current advances in understanding cellular network topology. Second, this chapter looks at the more specific problem of predicting which genes in an organism are critical to its survival, using only topological analysis of cellular networks.

7.1.1 Genetic Regulation Networks

The instructions for building proteins contained within a gene are in the form of sequences of pairs of the four DNA nucleotide bases: adenine, thymine, guanine, and cytosine. The process of transcription and translation convert a gene into its respective protein. First, RNA polymerase, an enzyme, binds to the promoter region on the DNA, while at the same time unzipping it and catalyzing the formation of a messenger RNA (mRNA) sequence. In the mRNA molecule, each base pair is complementary to the base pair on the DNA strand (so that cytosine still pairs with guanine and adenine pairs with uracil, the replacement for thymine in RNA molecules). After this mRNA sequence is spliced and processed, with a polyadenine sequence and a GTP cap added, the final mRNA molecule leaves the nucleus and goes to the translation step. At this step, ribosomal RNA (rRNA) and transfer RNA (tRNA) both transcribe the sequence of DNA into amino acids, which then fold in unique ways based on their physical and chemical properties.

The first step of this fairly complicated process, the binding of RNA polymerase (RNAP) to the DNA strand's promoter region to catalyze the formation of the mRNA molecule, is the one of interest here. The binding of the enzyme RNAP to the DNA strand can be affected by the presence of other proteins. Other proteins can inhibit (repress) the binding of RNAP to the promoter region of the DNA strand by blocking the region. On the other hand, other proteins can also activate the binding of RNAP to the promoter region of DNA by binding to regions of the DNA where they directly interact with and call RNAP, thereby facilitating the

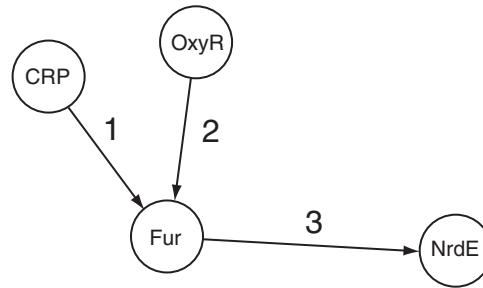


Figure 7.1 A protein regulation subgraph from the *E. coli* regulation network. This graph consists of four proteins—CRP, OxyR, Fur, and NrdE—as nodes and interactions as edges, that is, OxyR and CRP regulate Fur, which regulates NrdE.

translation of the associated genes. Thus, proteins can regulate the activity of RNAP, which governs the transcription of the DNA of other genes and subsequently the production of other proteins.

The interaction describing how one protein regulates the production of another via control of RNAP can be modeled as an edge in a graph. Remember, mathematically speaking, a graph is not the same as a chart (which includes bar charts and pie charts). A graph is a set of nodes (or vertices) with edges (connections) between them.

Figure 7.1 shows genetic regulation interactions among four proteins. A directed edge (an edge with an arrow) signifies that the source protein either activates or inhibits the sink protein. The graph shown in Figure 7.1 is actually only a very small part of a much bigger graph. Figure 7.2 displays the entire genetic regulation graph for *E. coli*. It is important to note that the graph describes only the end results of the regulation: the steps regarding the actual translation process are not notated. Instead, only the proteins that regulate the binding of RNAP to the DNA promoter region and the proteins eventually transcribed are included in the graph.

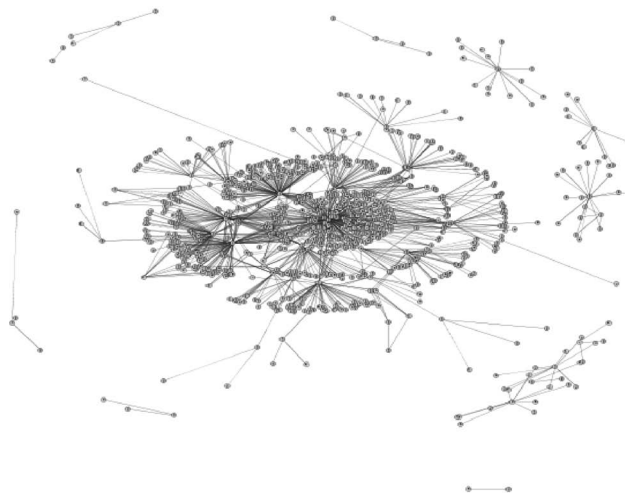


Figure 7.2 The *E. coli* genetic regulation network. Nodes are proteins and edges are regulations.

The data in this graph is obtained by researchers continually contributing to the curated EcoCyc database [1]. Generally, such data is originally gathered experimentally. It is possible to tell which proteins regulate others by measuring varying presence levels of each and performing basic statistical analyses to determine which genes influence the presence of others, especially when certain genes can be “turned off” using gene knockout. Gene knockout refers to radioactively causing a gene to not be expressed within a cell, and has been used on model organisms *S. cerevisiae* [10, 11] and *C. elegans* [12]. The creation of databases that profile organism genomes embodies one of the many goals of bioinformatics, which is to unify and universalize naming protocols and normalize data via large, nearly complete databases [13].

7.1.2 Protein-Protein Interaction Networks

An interesting graph can be formed by looking at the protein-protein interactions (PPIs) within a cell. PPIs are critical to a cell’s vitality. Unlike genetic regulation networks, a graph describing PPIs does not bypass any intermediate steps involving genes. Instead, a PPI graph describes actual interactions. That is, a PPI graph describes the binding of proteins to one another. Also, unlike with genetic regulation, which deals only with regulating how proteins should be translated, protein-protein interactions are important for the actual roles that the networks serve.

PPIs are the principal mechanism of cell communication. When hormones or other extracellular signals act as stimuli for a cell, the PPIs relay the message throughout the cell, instructing and catalyzing different organelles to respond appropriately [14]. The precise signaling mechanisms among these proteins are amazing—thousands of messages are carried through the cells in a human body every second, and hardly a single error is made. In fact, a cell whose protein-protein interactions simply vanished would almost immediately disintegrate, with the cell having no means of recognizing external factors, and no means by which to maintain something as basic as its cellular three-dimensional structure [15].

Figure 7.3 shows the entire PPI graph for *E. coli*, using the spring-embedded layout. (Spring-embedded layout is a format in which to visualize networks. Edges are treated as springs with a tension proportional to the length of the edge, and nodes are thought of as small magnets repelling each other. Then, starting from a random placement of the nodes in space, the nodes are allowed to move freely to relax such that the overall tension among the nodes is minimized [16, 17]. By changing the spring constants or the repelling forces, different layouts can be obtained.) Notice that there is a similar structure to both the PPI graph and the genetic regulation graph. Although the PPI graph is much larger, with almost two-and-a-half times more nodes (proteins) included, both have highly connected central proteins that radiate out to less centrally connected proteins. This is the cause of the scale-free property of such biological networks that is discussed later in this chapter. Topologically speaking, the structure shared by both the PPI network and the genetic regulation network is important in developing methods to analyze the properties of the proteins in the networks. Indeed, by exploiting the network properties of the graphs formed by protein interactions and regulations, computational meth-

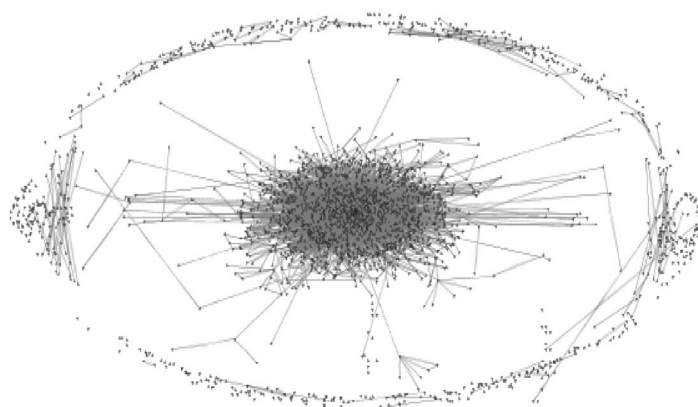


Figure 7.3 The PPI graph for *E. coli*, using the spring-embedded layout.

ods can achieve surprising accuracy in predicting biological consequences to gene disruption, as is also discussed later.

7.1.3 Metabolic Regulation Networks

All organisms, whether they ingest food or photosynthesize it, must utilize metabolic pathways in order to extract energy from macromolecules or build larger molecules. Each metabolic pathway starts with a metabolite, or relatively small molecule, and the metabolite is changed step-by-step along the pathway. Figure 7.4 displays a sample reaction. In *E. coli*, this network is a large one that describes all of the possible metabolic reactions in a cell.

At first glance, metabolic reactions pose a paradox. If metabolic reactions could occur spontaneously, any high-energy food would decompose within a fraction of a second. Surely, if this occurred, an organism would literally combust from within, with all of its energy-rich molecules being broken down and the energy within released all at once. On the other hand, if the conversion of energy-rich molecules (as in Figure 7.4) does not happen spontaneously (which it doesn't, since one can see a molecule such as glucose in a stable form even *in vivo*) then energy could never be extracted from ingested nutrients. The answer to the apparent paradox as to whether metabolic reactions occur spontaneously or not is that enzymes regulate

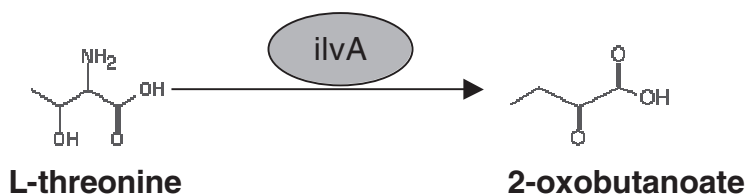


Figure 7.4 L-threonine is converted to 2-oxobutanoate by removing an ammonia molecule. Gene *ilvA* regulates the reaction by coding for the enzyme that actually catalyzes the transformation.

each step of every pathway. Proteins specifically designed to handle these metabolites facilitate their transformation at every step along a metabolic pathway. Metabolites fit into the active sites of enzymes. Often, an enzyme can only fit one specific metabolite in its active site, so the amount of end product produced is equally dependent on the substrate (metabolite) and enzyme. Varying amounts of metabolite have the capability to change the reaction rate. It is this fact that allows genetic regulation to play a role within metabolic regulatory networks, since varying levels of gene expression can impact the flow of metabolic activity through the metabolic network.

The nodes in metabolic networks are metabolites, and the edges correspond to reactants between metabolites. A more comprehensive description of metabolic networks takes into account the directedness of the edges as well: the edges start at reactants and point to products, designating the direction of the particular chemical reaction. Furthermore, metabolic networks allow for the introduction of the concept of *edge weights*, which are in general real valued numbers assigned to each edge in the network based on metabolite flux in reaction. In the case of metabolic networks, the weights are associated with the respective fluxes of the reactions that substrates are transformed into. While difficult to measure in practice because of the large variety of environmental conditions that influence the reaction fluxes, they have been estimated through the flux balance analysis method (FBA) [18]. Flux balance analysis is a technique to obtain the flux distribution in the system, subject to such optimization conditions as maximization of biomass production or minimization of nutrient utilization. It has been found that the distribution of fluxes in the metabolic network of *E. coli* grown in glucose follows a power-law distribution [18], similar to the scale-free property of the network topology itself, discussed in the next section.

7.1.4 The Scale-Free Property: A Network Characteristic

Before continuing this chapter's discussion of cellular networks, an important network characteristic must be analyzed. Many networks, biological and nonbiological, such as social networks and the World Wide Web (WWW) have been shown to be *scale-free* [19]. This means that a few nodes have a high number of links, while most nodes have very few. In general, a network is called scale-free when the probability $P(k)$ that a node has k links follows a power law, $P(k) \sim k^{-\gamma}$, where γ is the degree exponent and is constant for large k 's over an extended range of degrees. The *degree* or *connectivity* of a node is the number of other nodes that it is linked to in the network. The value of γ depends on the actual network, and in naturally occurring networks it is often $2 < \gamma < 3$.

One of the simplest and conceptually most seminal models giving rise to scale-free networks is the *preferential attachment* model [19]. This probabilistic model constructs networks by adding new nodes to the growing graph one by one, starting from a small connected core of a few nodes. Each time a new node is added to the network, it has to be linked to other nodes to ensure global connectedness; the number of new links that the node will use to make connections is constant, m . m can be chosen arbitrarily, but is usually a small number ($m = 1, \dots, 6$) due to the fact that the average degree of the nodes will be $\langle k \rangle = 2m$, and natural (biological) net-

works are usually sparse with a low average degree. The average degree is $2m$ since the total number of edges is mN with N being the number of nodes added to the system, and each edge contributes to the degrees of two nodes. The essential feature of the preferential attachment model is that each of the m edges of the new incoming node will be connected randomly to an existing node in the network, but the linking probability is higher for nodes that already have a large degree: it is linearly proportional to the degree of the target node. This phenomenon has been called the “rich get richer” phenomenon, since well-connected nodes will attract most of the new links to grow even denser. The degree distribution of the preferential attachment model can be calculated exactly: if an existing node i has degree k_i , and the actual size of the network is t at time t (there is one new node added in each time step), then by adding a new node, the expected increase in the degree of node i can be approximated by the following continuous rate equation:

$$\frac{\partial k_i}{\partial t} = m \frac{k_i}{2mt}$$

The left-hand side is the average increment of the degree for node i during this time step, and the right-hand side expresses the fact that each of the m new links have a probability linearly proportional to the degree k_i to increase it. $2mt$ is the sum of all degrees up to the time t and provides the proper normalization for the connection probability. Note that the increase as described above is a fractional number and can be thought of as the average increase over many runs of the model for node i . The solution of this equation is that

$$k_i(t) = m \left(\frac{t}{t_i} \right)^{\frac{1}{2}}$$

where t_i is the time of introduction (actual network size at that point) of node i , and the solution can be checked by substituting $k_i(t)$ back into the rate equation above. From this, it is possible to derive the degree distribution of the model network, which results in $P(k) \sim k^{-3}$ [19]. This means that the associated degree exponent is $\gamma = 3$.

This type of network is in contrast with the random networks provided by the Erdős-Rényi model that is considered to be the null model for random networks [20]. Given a fixed number of nodes, this model assumes that for any pair of nodes, the probability that a link is formed between them is a constant p , so that the number of links per node will follow a Poisson distribution peaking at $\langle k \rangle = p(N - 1) \approx pN$, the average number of links per node, if N denotes the number of nodes in the graph. Poisson-distributed networks are clearly homogenous with a uniform degree prevailing for most of the nodes, while scale-free networks are highly heterogeneous, with many nodes having only a small number of connections, but where a few highly connected nodes can also be found in nonvanishing numbers.

A model motivated by evolutionary principles is the *duplication-divergence model* of protein interaction networks. This is a growing model also; new nodes are being added to the network similarly to the preferential attachment model. Mimicking gene duplication, the new node i' will be a copy of a randomly selected node

i , which means that i' will be connected to all of the neighbors of i , reflecting the idea that the new protein is identical to the old one, and so can interact with the exact same proteins. Also, with probability p that is a parameter of the model, i' will be connected to i as well, since identical proteins may just as well interact. In the divergence step, for each of the nodes j linked to both i and i' , one selects either the link between i and j or i' and j , and removes it with probability q . This step corresponds to a random mutation in i or i' that alters their interactions (the divergence step). By choosing p and q appropriately, Vázquez et al. reproduce the degree distribution of the protein-protein interaction network of yeast with a high degree of accuracy [21]. The duplication-divergence model is an example of the preferential attachment mechanism in protein networks, since neighbors of highly connected proteins are more likely to be selected for duplication than the neighbors of proteins with a smaller degree; thus well-connected proteins have a larger probability of acquiring new interactions, giving rise to a scale-free degree distribution.

Despite the models being necessarily simplified representations of the in vivo processes, experimental results confirm the scale-free degree distribution postulated by many of these. Jeong et al. found that the connectivity distribution of the protein interaction network of *S. cerevisiae* is best described by a power law with an exponential cutoff [22],

$$P(k) \sim (k + k_0)^{-\gamma} e^{-(k+k_0)/k_c}$$

where the best fit was obtained by setting $\gamma = 2.4$, $k_0 = 1$, and $k_c = 20$ (see Figure 7.5). Note that any practical model of scale-free networks has to introduce a *cutoff* for the node degrees; in systems of finite size, the degree of the maximally connected

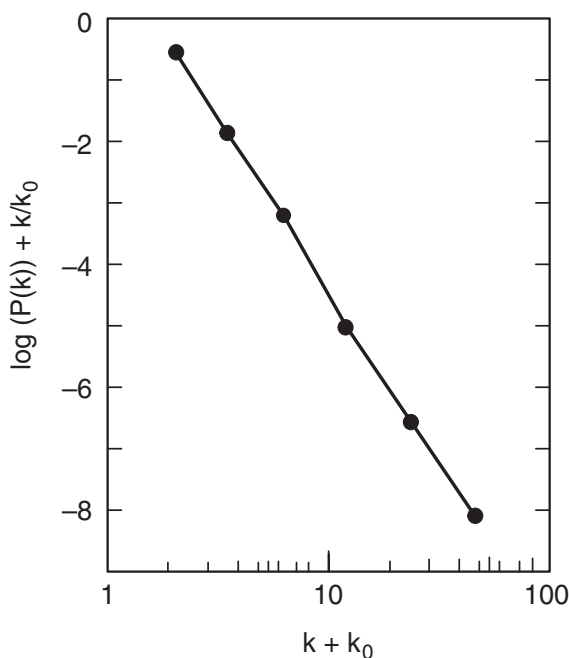


Figure 7.5 The degree distribution $P(k)$ of the protein-protein interaction network of *S. cerevisiae*. The experimental data have been fitted by a power-law function with an exponential cutoff, as described in the text (originally presented in [21]) (From Nature, Vol. 411, p. 41.).

node has to be bounded as well. The so-called natural cutoff for networks with power-law degree distributions follows from the condition that one has to find at most one node with or “above” the largest degree given by the unbounded power-law degree distribution function [23]:

$$N \int_{k_c}^{\infty} P(k) dk \approx 1$$

Here N denotes the number of nodes in the network again, and k_c is the expected cutoff degree. In the case of scale-free graphs, $P(k) \sim k^{-\gamma}$, and

$$k_c(N) \sim N^{1/(\gamma-1)}$$

This indicates that the degree of the most connected node grows slower than linearly (sublinearly) with the size of the network in practically relevant networks.

Although no single definition is accepted as a universal, formal definition for scale-free networks, Li et al. defined a scale-free metric [24] in the following manner: Let the set of all edges in a graph g be given by E . Then, define $s(g) = \sum k_i k_j$, for $(i, j) \in E$, where k_i is the degree of node i . Then, define $S(g) = s(g)/s_{\max}$, where $s_{\max}$ is the “maximum value of $s(h)$ for h in the set of all graphs with an identical degree distribution to g .” $S(g)$ will always fall between 0 and 1. Values for which $S(g)$ are close to 1 correspond to scale-free networks. Indeed, for most biological networks, studies have found that $S(g)$ is very close to 1.

The biological need for scale-free networks has been realized through natural selection. Scale-free networks make for quick communication, since the shortest path between any two nodes is minimized when there are hubs that are connected to many nodes. For example, if a membrane protein needs to communicate with a protein in the interior of the cell, it is much easier if the membrane protein can interact with a topologically central protein and for that protein to interact with the interior protein rather than require a lengthy pathway. Furthermore, scale-free networks have been shown to be robust and error tolerant, as seen later in the chapter. It is the scale-free structure of cellular networks that allows cell signaling and communication to occur at such an astonishing rate with high accuracy.

7.2 Topology of Cellular Networks

7.2.1 Network Motifs in Genetic Regulation Networks

Extensive past research has been performed on the large-scale and small-scale connectivity of various cellular networks. Balázsi et al. have shown that networks in both *E. coli* and *S. cerevisiae* are characterized by a lack of feedback loops, except in the case of direct self-regulation [25]. Self-regulation is achieved when the production of one protein from its corresponding gene prevents that gene from being transcribed and translated into more of its protein. In this way, cells can quickly and easily moderate the amount of certain proteins that are produced.

On the other hand, some small-scale motifs have been shown to be overrepresented in the cellular networks of *E. coli* and *S. cerevisiae* (as compared to random networks). Examples of small-scale motifs include the bifan motif and feed-forward

loop. The feed-forward loop is in contrast to the feedback loop in that in its structure, protein A regulates the production of protein B, and both proteins A and B regulate the production of protein C. The bifan motif is similar: proteins A and B each regulate both proteins C and D. Both motifs are displayed in Figure 7.6. Balázsi et al. also explain the ubiquitous presence of these motifs in terms of their information-handling abilities, as well as the aggregation of these motifs into large-scale topological structures.

The Kashtan et al. study of “motif generalizations” defines the families of motifs of different sizes, united by a structural theme, in genetic regulatory networks [26]. Although the current study of network motifs is hampered by the large computational time of finding motifs consisting of more than seven nodes, small motifs and their properties are being investigated. Kashtan et al. specifically generalize three different network motifs: the feed-forward loop, the single-input module, and the bifan motif. The authors define the number of *roles* in a subgraph as the number of permutations of nodes that preserve the structure of the graph. By duplicating the roles of nodes from the three-node original motifs, the authors define larger versions, or generalizations, of their motifs. Interestingly, the authors find that, although in most cases, generalizations of network motifs exist as subgraphs within the networks, the same generalizations about a network motif do not necessarily occur in different networks, even if the different networks all display that motif.

Balázsi et al. have found that since transcriptional regulation networks do not contain cycles except for self-regulation, the nodes of such a network can be categorized in a discrete number of regulatory layers. Due to such results, they conclude that regulatory-level cells (in the regulation-layered hierarchy) perceive their environment before conveying information by first dissecting complex signals into simpler perturbations processed by individual subnetworks rooted at the top of the hierarchy, those that have no incoming edges.

In a similar vein, Shen-Orr et al. studied three particular motif structures in the directed transcriptional regulation network of *E. coli* and found that their prevalence is statistically significant when compared to the null hypothesis, which they chose to be the randomly rewired version of the original regulation network [27]. By appropriately crafted algorithms they enumerated the three kinds of motifs and represented the regulation network in terms of these. The particular choice of motifs was made so as to emphasize the roles these motifs play in a signal-processing circuit: for instance, one of them, the coherent feed-forward loop, responds to persistent signals and rejects any signal that is present on a transient basis. This way, feed-forward loops can act as filters against quick fluctuations in the input signals. The motif representation of the regulation network simplifies the structure consid-



Figure 7.6 The feed-forward loop (left) and bifan motif (right). Arrows represent positive or negative regulation.

erably and allows for the description of its modules as logical units in a manner similar to the approach used in systems control theory.

It is important that this work and these conclusions are applied only to transcriptional regulation networks, which are directed. The motifs utilized depend on the directionality of regulation interactions, which are lacking in protein-protein interaction networks.

7.2.2 Topological Characterization of Protein Networks

Protein-protein interaction networks contain much information about cellular organization within their topology. The interactions in which a protein is involved often correlates with its functional characteristics. For this reason, protein-protein interaction graphs are often analyzed in order to elucidate the function of unstudied proteins [28, 29]. For example, proteins with similar functions are often closely clustered in protein-protein interaction networks. This is fairly intuitive, considering the use of such networks within a cell. Biologically, these networks serve to communicate information. Thus, a protein is likely to bind to functionally close proteins in order to convey information regarding its function. This is made possible by the high level of specialization in proteins.

The notion of this clustering is reinforced by the concept of modularity. A *module* is a group of proteins that cooperate to carry out a well-defined, single biological goal [30]. It is well-established through experiments that many biological functions are carried out by protein modules, even though it is unlikely that all functions require such groups [31]. Intuitively, such modules should be present as highly connected clusters with few links outside of the module. However, the scale-free nature of protein-protein interaction networks demands that the entire network be dominated by a few nodes with a large number of links, with the rest of the nodes having few links. Thus, the requirement that hub proteins must have a large number of links prevents isolated modules from existing [31]. Yook et al. proposed that hierarchal modularity can reconcile the scale-free structure of the network with the biologically suggested principle of modularity. Hierarchal modularity is similar to the layered hierarchy used to describe transcriptional regulation networks. Modules are formed by closeness within a hierarchal level, while the scale-free structure is still preserved by having a few root nodes at the top of the hierarchy, as in the case of the transcriptional regulation network. Clusters form while still linking to highly connected nodes.

The network topology of protein-protein interaction networks features some characteristics common to transcriptional regulation networks as well. Both types of networks are usually comprised of one large component with several, if not many, isolated islands of interconnected nodes that do not connect to the central component. In the spring-embedded layout, the large central component of such networks often has a very small radius.

The use of network topology in studying protein-protein interaction networks has proven very fruitful. Indeed, some important biological conclusions have been drawn from computational studies of protein network topology. For example, much of the scale-free structure in such networks has been attributed to the fact that hubs allow for quick communication among proteins. Computational studies have

revealed that proteins specialized for cellular communication and signal processing are the most segregated, with the most pronounced modular qualities.

Yook et al. also considered one more interesting use of protein network topology. They divided proteins into 28 spatial compartments within a eukaryotic cell. They were then able to analyze the correlation between network topology and cellular localization. The premise of this study is that a protein found in a certain region is much more likely to interact with other proteins from the same region than proteins found in other spatial regions. The authors provided as an example the hypothesis that a nuclear protein is much more likely to interact with another nuclear protein than it is to interact with a protein found in the cell wall. The authors found a very interesting phenomenon. Their hypothesis was correct, and topologically close proteins were also spatially close. However, unexpectedly, this correlation was even stronger than the correlation between network topology and protein function (instead of location).

7.2.3 Topology of Metabolic Networks

Metabolic networks are inherently different from genetic regulation and protein-protein interaction networks in that they participate in the transformation of small chemical compounds instead of the direct interactions among proteins and genes, which are characteristically complex molecules. It is important to note that metabolic networks still include gene involvement, since every metabolic interaction (from one metabolite, or small compound, to another) is regulated by at least one enzyme, which is a protein with a corresponding gene.

However, even though metabolic networks are very different from other types of cellular networks, they have been shown to sometimes display a higher degree of topological organization [32]. Jeong et al. found that the topology of metabolic networks closely resembles the organization of nonbiological systems [32]. This may indicate that metabolic organization is nearly identical among all living organisms and that it obeys the principles of certain types of general networks as well.

Studies that have graph-theoretically modeled metabolic networks have shown that they, like the other cellular networks, conform to a scale-free structure. This means that the probability distribution function for the number of links a node has closely follows the power-law distribution $P(k) \sim k^{-\gamma}$. For example, for the *E. coli* metabolic network, the power-law distribution holds for $\gamma = 2.2$ [32]. However, unlike in the cases of the genetic regulation network and the protein-protein interaction network, the scale-free structure has little to do with signal processing or intracellular communication. Instead, metabolic networks may be scale-free because certain metabolites are much more highly used in a cell than others due to the variable versatility of different substrates. Compounds that are easily mutable by catalytic enzymes are likely to participate in more interactions than compounds that take part only in very specific biochemical reactions. Apparently, this scale-free structure describes the metabolic networks of numerous studied organisms from all three domains of life, corroborating the notion set forth by Jeong et al. that the metabolic networks' similarity to nonbiological systems may imply a profound simplicity in metabolic networks so that such networks belonging to all organisms are similar [32].

This original premise is extended further by a study of the metabolic *network diameter*. For any network, the network diameter is defined as the average shortest path between all pairs of nodes. Thus, homogeneous networks such as the random networks described by the Erdős-Rényi model will have a large network diameter, while scale-free networks will maintain a smaller diameter. However, it has been found that the network diameter for the scale-free metabolic networks is constant despite the size of the network, which differs from organism to organism. This means that as new substrates are added and metabolic networks grow increasingly complex in more complex organisms, substrates grow increasingly connected.

Jeong et al. also studied the robustness of metabolic networks in preserving their relatively low network diameter. They performed random error simulations on the metabolic network to determine if the network would break down in the case of enzyme mutations. In a metabolic network, an enzyme mutation is tantamount to removing an edge, since enzymes are essential in forming the edges between metabolites. Random errors were simulated by removing random links and substrates, creating the same effect as mutations in catalytic enzyme formation. Network robustness was measured in terms of the network diameter. Researchers found that the random removal of substrates did not change the network diameter. This has been corroborated by *in silico* and *in vivo* studies in which enzyme mutation revealed fault toleration in *E. coli* [32].

The topology of the metabolic network can also be considered in terms of the actual substrates. Indeed, it appears that even though only 4% of all substrates appear in common among 43 different organisms surveyed, the most highly connected substrates among all of these organisms are nearly identical [32]. This leads to the biological conclusion that the roles of very important substrates have been conserved during the evolution of various organisms. This is just one example of how computational studies can lead to biological conclusions that experimental biology can sometimes never achieve.

Ravasz et al. extended the concept of modularity that was first discussed earlier in this chapter's treatment of protein-protein interaction networks to metabolic networks as well [33]. Modularity is the formation of modules that contain groups of substrates with a close spatial or functional proximity. Ravasz et al. studied hierarchical modularity using *clustering coefficients*. The term clustering has been borrowed from the social sciences, but its meaning has become somewhat narrowed to refer to the prevalence of connected triangles in the network. In other words it is a commonly observed feature in many real networks that if a node is connected to either node of a connected pair of nodes, then it is likely also connected to the other node of the pair. The clustering coefficient of a node i is the proportion of links that exist between neighbors of node i divided by the number of links that could potentially exist between neighbors of node i :

$$C_i = \frac{n_i}{\frac{k_i(k_i-1)}{2}}$$

where C_i is the clustering coefficient of node i , n_i is the number of connections between neighbors of this node, and k_i is its degree. Note that the denominator above yields the maximum number of possible links between neighbors, as described by

the definition of the clustering coefficient. C_i always falls between 0 and 1. Thus, a node with a high clustering coefficient is surrounded by neighbors that are all connected to one another, forming the smallest unit of a topological module. Large values of the clustering coefficient can be an indication of hidden modularity in the network, especially when the network itself is the representation of certain functions, as is the case for example in metabolic networks [33]. Modularity is inherently subject to a variety of interpretations, due to the many degrees of freedom that can go into designing the algorithms that classify the nodes into modules [34]; the clustering coefficient inside modules can still be expected to be considerably higher than in intermodular regions. Figure 7.7 calculates clustering coefficients for a few sample networks. It has been found that the average clustering coefficient for metabolic networks is much larger than the expected average clustering coefficient for scale-free networks of comparable size. This indicates that there is a disproportionately high amount of modularity in the metabolic networks. However, this poses the same problem as encountered in the case of the protein-protein interaction networks. The existence of modularity, as predicted by the clustering coefficient test, is at odds with the power-law distribution model for scale-free networks. Obviously, if a few hubs are to dominate the network with most of the nodes connected only to one of these hubs, modules do not have the capability to form.

The solution to this is a model of hierarchical modularity as discussed before. Ravasz et al. offered a generating heuristic to simulate metabolic networks. In this heuristic, a densely connected module of four nodes is created. Then, this module is replicated three times, and each of the 12 newly generated nodes is connected to the central node of the original cluster. This process is iterated until the desired network size is achieved.

The model does surprisingly well. By heuristically generating this network type, which integrates modularity into a scale-free network, Ravasz et al. achieved excellent results. When comparing this model network with the real metabolic network of *E. coli*, researchers found that the degree exponent value was 2.26, very close to the 2.2 value found in actual metabolic networks. Furthermore, the clustering coefficient was approximately the same as well, 0.6. Ravasz et al. also noted that the clustering coefficient was independent of the size of the network, as was the case when comparing metabolic networks of organisms with different numbers of substrates. Note that because this model involves an iterative procedure that adds

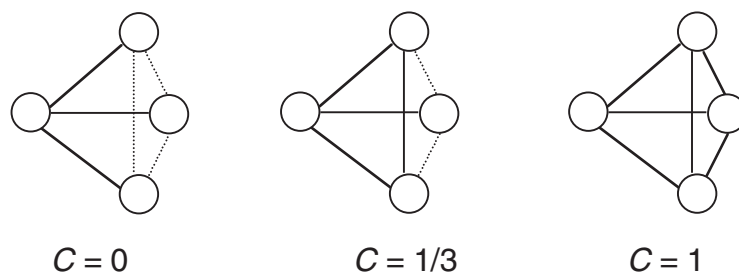


Figure 7.7 Clustering coefficients are calculated for the node on the left-hand side. Solid edges represent actual connections, while dashed edges represent potential connections. In the first cluster, none of the left node's neighbors share edges, so there are zero out of a possible three connections. In the second cluster, one of the possible three connections exists, so the clustering coefficient C is $1/3$. In the third cluster, all three possible connections exist, so C is 1.

layers to reconcile modules with hubs, the modules have hierarchical organization. The fact that the modeling heuristic produces a nonbiological network that closely mimics real metabolic networks in terms of clustering and the power-law distribution of node degree implies an elegance in metabolic networks since their behavior can be computationally approximated.

A gene is considered an *essential gene* if an organism dies without its expression. Currently, there are a number of experimental methods to determine which genes in an organism are the most important, for example, radioactive gene knockout. Typically, radioactivity is used to remove a gene from an organism's genome completely. Then, when the organism reproduces, its offspring do not have the gene either. In all of these progeny, the removed gene is not expressed, meaning the effects of its corresponding protein are not realized. If the organism then dies, then the protein was essential to the organism's survival, so the gene is an essential gene.

The motivation for finding essential genes through computational methods is twofold. Although researchers have identified essential genes in some organisms via testing each gene individually through gene knockout techniques, such methods are expensive, time consuming, and offer less insight into the basic workings of the cell than do computational methods [35]. Secondly, computational techniques can reveal aspects of gene essentiality in organisms that may be too complex or too rapidly changing to effectively study experimentally. Additionally, the computational study of essential genes holds interest because it can be used to study a cell's minimal genome, or the minimal set of genes required for the cell to survive and function [36].

So why use cellular networks to predict essential genes if there are plenty of experimental methods that achieve the same end? The problem with clinical experiments is not only that they are time consuming, expensive, and limited only to the tested organism, but that they do not reveal a great deal of information about other biological profundities in the organism [35]. In the case of experimental methods, all that is being discovered is the end result of removing a gene from the genome. With computational methods, it is possible to gather the information in a systematic way to see more clearly the specific mechanisms that are causing the cell to die, whether it is due to a missing link in an important communication chain or misregulation in the production of an important metabolite [37]. Furthermore, computational methods are quicker and cheaper, so they can be applied to many organisms for which experimental methods may be too cumbersome to perform [36]. Finally, one major area of interest to many bioinformaticians is the human genome. However, experimental studies on human gene essentiality are clearly not practical, since such tests usually require the fatality of many specimens. Thus, with the computational study of model organisms such as *E. coli* and *S. cerevisiae*, aspects of human gene essentiality can be studied as well. There are a number of computational methods to analyze cellular networks and predict which genes in an organism are essential. The advantage here is that such computational approaches can be used as a filter to help constrain the biological testing. Biological testing for the essentiality status of specific genes can be lengthy and organism-dependent. For example, certain techniques such as RNA [38] and transposon-generated knockouts [39] are not suitable for lower organisms like *E. coli* but can be used in others, such as mice.

7.2.4 Adjacency Matrices

Before continuing the discussion of gene essentiality prediction algorithms, it’s important to note *how* these matrices are stored and analyzed. Most of the data and algorithms discussed in this chapter were tested and developed in the Matrix Laboratory (MATLAB). However, it is not practical to store the actual graph and perform algorithms upon the graph. Instead, a more concise and mathematically accurate method is needed. Graphs are stored as adjacency matrices. An adjacency matrix that represents a graph with n nodes is an $n \times n$ square matrix with each element $(i, j) = 1$ if the node i activates, interacts with, or transforms into node j in the genetic regulation network, PPI network, or metabolic pathways network respectively. In the case of the genetic regulation network, $(i, j) = -1$ if node i corresponds to a protein that inhibits the protein that node j corresponds to. If nodes i and j share no relationship in any of the three networks, then $(i, j) = 0$. A sample adjacency matrix and corresponding graph are shown in Figure 7.8.

7.2.5 Hubs

Jeong et al. [22] used the hubs method to predict essential genes in protein-protein interaction networks. This method counts the number of neighbors each node has. Then, nodes that have the most neighbors are predicted to correspond to the most essential proteins and genes. This method is founded upon the idea that the most centrally located proteins in a PPI network are involved in the most signaling pathways. Thus, when these highly connected proteins are removed, many pathways are destroyed, hindering cell functionality. This culminates in cell death for many cases. Further studies have shown that this method also performs well in genetic regulation networks, although essential gene detection rates are significantly lower [37].

One hypothesis as to why the detection rate is lower in regulatory networks is that the removal of proteins critical to PPI networks leads to a handicap for the cell in terms of communication and function, while the removal of a critical regulator does less to harm the cell since some genes are simply over- or underexpressed, which does not lead to cell death in as many cases. Although the hubs method provides a good way to determine essential genes, it is even more important to note that by using computational methods to analyze gene essentiality, it is possible to formulate meaningful and interesting hypotheses about the role of proteins that would have been difficult to formulate otherwise [40]. Had research on gene essentiality been limited only to experimental analysis, the observation that the most involved proteins are the most essential would have never been made.

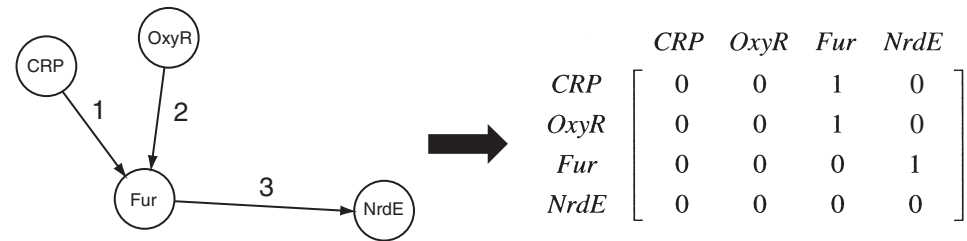


Figure 7.8 The graph in Figure 7.1 represented as an adjacency matrix.

The benefit of the hubs method is that it provides fairly good results and has a quick runtime. Also, it is fairly simple and easy to implement, providing a useful tool for many researchers. One drawback is that it is fairly limited in scope. One hypothesis generated by the moderate success of the hubs method is that other methods that look past simply neighbors will be more accurate in predicting essential genes than the hubs method [37]. The results of Jeong et al. exceed 40% detection rates only in the PPI network, but are much lower in the genetic regulation network. Thus, the challenge is to not only find a method that serves well in both types of networks, but also one that will give higher essentiality detection rates.

While it is an established fact that hubs play a very important role in predicting gene essentiality, a higher level of description has to take into account the topological and dynamical position of the hubs in the PPI network. Han et al. characterized hubs based not only on the number of their interaction partners, but also on the time when they interact with these proteins [41]. They classified hubs as “party” hubs that interact with most of their partners simultaneously, and “date” hubs that bind their different partners at different times or locations. They found that both party and date hubs are essential approximately to the same amount, but their explanation for this essentiality is different for the two classes: party hubs are the central mediators inside local modules, so their removal disrupts important specialized functions. On the other hand, they suggest that date hubs serve as global communicators between multiple modules and thus are crucial for robustness from this perspective.

7.2.6 Reachability

One method to predict essential genes stems from the concept of reachability, introduced by Alterovitz et al. [40]. A *reachability matrix* is defined as an $n \times n$ matrix that represents a graph with n nodes, but with an element $(i, j) = 1$ if there exists at least one path from node i to node j , and $(i, j) = 0$ otherwise. The reachability matrix of a graph is calculated by simply raising the adjacency matrix of the graph to powers of k for $k = 1, 2, \dots, n$, and recording $(i, j) = 1$ if the element (i, j) is nonzero for any value of k . A *reachability graph* is the graph that is formed when the reachability matrix is represented in node-edge format. Further, the *reachability index* of a graph is the sum of the elements of the reachability matrix or, equivalently, how many edges are present in the reachability graph, which yields the number of paths that exist between all combinations of pairs of nodes in the graph. Figure 7.9 shows the reachability matrix of the graph represented in Figure 7.8. Its reachability index is the sum of the elements of the matrix (the number of 1's), or 5.0.

Reachability indices give rise to another method to predict essential genes. In the reachability method, the removal of the n proteins represented in the graph is simulated by deleting them from the graph. Then, the reachability matrix and corresponding index is calculated for each protein's removal. This method was developed to try to account for flaws in the hubs method. For example, the hubs method looks only at proteins that are themselves highly connected, but does nothing to account for proteins which aren't highly connected themselves but link two other highly connected proteins. Clearly, such proteins are important in signaling pathways and in genetic regulatory networks, but would not be counted since they may

	<i>CRP</i>	<i>OxyR</i>	<i>Fur</i>	<i>NrdE</i>
<i>CRP</i>	0	0	1	1
<i>OxyR</i>	0	0	1	1
<i>Fur</i>	0	0	0	1
<i>NrdE</i>	0	0	0	0

Figure 7.9 Reachability matrix corresponding to the graph and adjacency matrix in Figure 7.6.

have as few as two neighbors, whereas some proteins have more than one hundred. In the reachability method, nodes which, when removed, reduce the reachability index of the graph the most are predicted to be essential.

Since proteins that lie in between two highly connected proteins often serve as unique pathways, their removal is often very pronounced through the reachability method. Unfortunately, this method has not done well in preliminary tests, which have been limited due to its fairly lengthy runtime. However, the method holds promise for the future as other techniques are incorporated in order to refine current algorithms. For example, with consideration of the functionality of certain genes, the reachability method can be modified to remove pairs or trios of functionally related genes and calculate reachability indices of all such combinations. This modification is based on the concept that there may be groups of genes so functionally similar that the deletion of just one does not have any impact on the cell in terms of vitality. Instead, the removal of more than one, or perhaps all, of the genes in the group may lead to cell fatality.

Future modifications to the reachability method also must include aspects to shorten the runtime. The runtime of the reachability matrix is given by $O(n^5)$, where n is the number of nodes in the graph. This is because the matrix multiplication of an $n \times n$ matrix with itself has an order of n^3 , and for each of n nodes, this multiplication must be performed n times. In order to incorporate larger networks, the reachability method must be also run on faster computers, since the runtime increases greatly for large graphs.

7.3 Gene Ontology and Functional Clustering of Essential Genes

Gene Ontology (GO) is a controlled vocabulary to describe gene and gene products in any organism [42]. It is a directed, acyclic graph, with terms and categories falling under one another. The three organizing principles are molecular function, biological processes, and cellular component. GO describes how genes and gene products behave in the context of the cell. A group of genes is *GO enriched* in a particular category if a significantly larger proportion of the genes from that group occur in that category than would if the group were a completely random selection from the organism's genome. Essentially, GO enrichment refers to an overrepresentation of GO terms associated with a particular biological process, cellular component, or molecular function. For example, if a group of genes is found to share a unique topological property, such as belonging to the same cluster, and 40% of the

genes in this group are related to the electron transport chain step of photosynthesis, whereas only 3–4% of all genes in the organism correspond to that function, then the group of genes is GO enriched in the GO category corresponding to the electron transport chain. This is because the group of genes overrepresents the genes related to the electron transport chain mechanism. Keep in mind that the number of genes in the group must be fairly large so that the difference from a random group of genes displaying the same proportions is statistically significant. The significance is calculated via the following formula:

$$p = 1 - \sum_{i=0}^{k-1} \frac{\binom{K}{i} \binom{N-K}{n-i}}{\binom{N}{n}}$$

where p is the p-value, n is the number of genes in the group, k is the number of genes from the group that fall within the GO category of interest, K is the number of genes in the possible set of genes, and N is the number of genes in the genome, from which we take the n .

Functional clustering is an interesting feature of essential genes. However, previously the existence of functional modules was known to apply only to the entire proteome. Alterovitz et al. showed that essential genes form functional clusters as well. Two remarkable characteristics about the topological properties of essential genes in both organisms are based on the concept of functional clustering.

Figure 7.10 shows the graph of only essential genes in both the genetic regulation network and the PPI network in *E. coli*. In both, there are two clearly defined circled clusters. This is the first key characteristic of essential gene topology. A cluster here is a group of genes that are each highly connected with other genes in the cluster, but have very little connectivity outside of the group. The fact that such clusters form is interesting. If essential genes were randomly scattered, then such clusters would not form. Also, not all essential proteins in the proteome are a part of a cluster. In fact, less than half are [37, 40]. It is currently not known if there is a fundamental attribute or property of the genes in the clusters that separates them from the others. Also, it is noteworthy that there are precisely two clusters in every essential gene-only graph, although this may not have a biological significance.

The second key characteristic about essential gene topology is more interesting and useful: each cluster is GO enriched for at least one category. This means that the clusters are grouped according to function or role in the cell. For example, one of the clusters in the genetic regulation network in *E. coli* is enriched for structural roles ($p < 6.00 \times 10^{-5}$) [40]. Thus, the genes in that cluster all tend to be involved in providing structure to the cell. Another example comes from the protein-protein interaction network, in which one of the clusters is highly enriched for controlling the cell cycle, cell metabolism, and protein biosynthesis ($p < 4.0 \times 10^{-11}$) [37]. The fact that essential gene clusters are GO enriched is extremely significant. One application of this property is that clusters of essential genes in organisms can be identified, and then, since most organisms do not have complete lists of essential genes (many genes remain untested for logistical purposes), an essential gene graph is

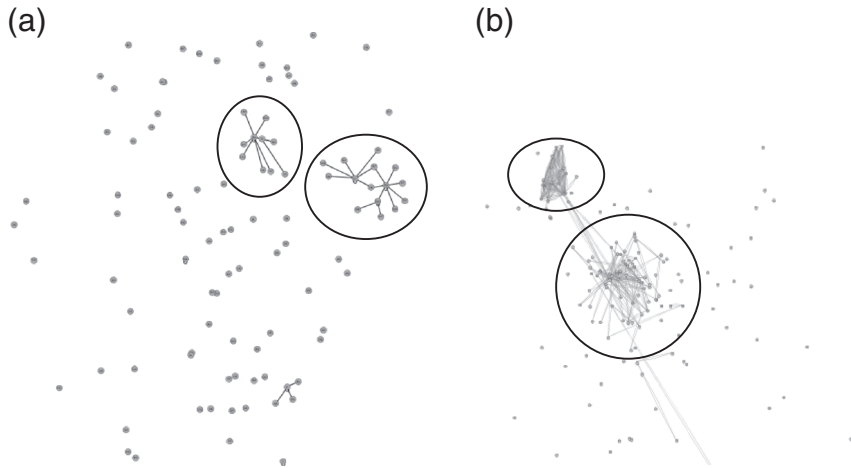


Figure 7.10 Graphs of only essential genes. In (a), the genetic regulation network, two clusters form, of 10 and 14 genes. In (b), the interaction network, two clusters also form, this time of 14 and 44 genes. In both cases, the rest of the graph remains almost entirely unconnected.

grown. What this means is that enrichment probabilities for various GO categories are calculated, and then genes with unknown essentiality status that most closely fit these GO categories can be tested first. Often, the reason the testing of essential genes in organisms is incomplete is that monetary and logistical limitations permit only a small number of genes to be tested. By intelligently selecting essential gene candidates, fewer resources will be needed to complete essential gene lists.

Another application of the property is to determine which functions and processes in a cell rely most critically on the expression of certain genes. Most processes that occur within a cell are critical, but since roughly only 10% of an organism's genome consists of essential genes, most of the genes that aid in these processes are not on their own essential. However, essential genes that functionally cluster all carry related roles in the cell and, since they are essential, the corresponding cellular function is most precarious, perhaps because the function is extremely important, such as the regulation of mitosis.

The property of cluster enrichment points to a deeper elegance in essential genes than before seen. Functional clustering means that genes with similar function tend to regulate each other and bind together more often than not. Signaling pathway proteins must be topologically close to proteins of similar function. Thus, the conclusion is moot in the case of the PPI network. But the fact that genetic and transcriptional regulation occurs in functional clusters is one only elucidated by the computational analysis of essential genes, which again applies to the broader principle motivating computational gene-related studies.

Moreover, although the essential genes involved in the *E. coli* metabolic network (essential genes that regulate at least one reaction) do not functionally cluster, the entire set of essential genes, which includes 23 genes, is itself highly GO enriched for oxidoreductase activity, acting on the aldehyde or oxo group of donors, with disulfide as an acceptor ($p < 1 \times 10^{-3}$) [37]. This fact, like GO enrichment in the other two cellular networks, can be exploited to increase essential gene detection rates in future methods.

7.4 Conclusion and Future Avenues

This chapter has covered a few interesting properties about biological network topology. Not only are such networks scale-free, their form follows their function. The treatment of the three main biological networks here has shown the variety of methods used to predict essential genes in organisms. Future avenues that current researchers are taking include generalizing the results discussed in this chapter. Although it is a good model organism, results and conclusions obtained using *E. coli* can only be meaningfully generalized to bacteria. However, some of the most important organisms to humans are eukaryotes. Thus, it is important to extend more research on computational gene essentiality detection to more eukaryotic model organisms.

Also, as far as essential gene detection goes, future research must undoubtedly include refining current methods to improve detection rates. Integration of GO in essential gene prediction is the most probable avenue of study. Currently, GO enrichment analysis has been performed on essential gene clusters, but the results of this analysis have not been applied in turn to improve essential gene detection. As suggested in the previous section, GO can be a useful tool in selectively considering genes and clusters of genes when looking to test a few for essentiality. Even if GO does not lead to the improvement of computational methods, the least it can do is limit the number of genes that must be experimentally tested. Furthermore, the phenomenon of functional clustering begs more research. If functional clustering is indeed a phenomenon present in many organisms, and if, even more surprisingly, essential gene networks tend to form exactly two functional clusters, many questions about the evolution of such phenomena are raised.

Another avenue of research that will surely yield powerful conclusions once completed is effectively combining the PPI network and the genetic regulation network. Unfortunately, current data exists only for each network such that the intersection of the two contains insignificantly few nodes. In the future, if both graphs can be extended to include a larger percentage of the 4,000 gene-plus genome of *E. coli*, studies that consider both protein interactions and protein transcriptional regulations can be used to develop even more sophisticated theories and models.

References

- [1] Karp, P. D., et al., "EcoCyc: an encyclopedia of *Escherichia coli* genes and metabolism," *Nucleic Acids Res.*, Vol. 24, 1 Jan. 1996, pp. 32–39.
- [2] Edwards, J. S., R. U. Ibarra, and B. O. Palsson, "In silico predictions of *Escherichia coli* metabolic capabilities are consistent with experimental data," *Nat. Biotechnol.*, Vol. 19, Feb. 2001, pp. 125–130.
- [3] Martin, M. J., et al., "Comparing bacterial genomes through conservation profiles," *Genome Res.*, Vol. 13, May 2003, pp. 991–998.
- [4] Tugendreich, S., et al., "Alu sequences in RMSA-1 protein?" *Nature*, Vol. 370, 14 July 1994, p. 106.
- [5] Hieter, P., D. E. Bassett, Jr., and D. Valle, "The yeast genome—a common currency," *Nat. Genet.*, Vol. 13, July 1996, pp. 253–255.

- [6] Jordan, I. K., et al., "Essential genes are more evolutionarily conserved than are nonessential genes in bacteria," *Genome Res.*, Vol. 12, June 2002, pp. 962–968.
- [7] Kamvysselis, M., "Computational comparative genomics: genes, regulation, evolution," in *Department of Electrical Engineering and Computer Science*. Ph.D. thesis, Cambridge, MA: MIT, 2003.
- [8] Seoighe, C., and K. H. Wolfe, "Yeast genome evolution in the post-genome era," *Curr. Opin. Microbiol.*, Vol. 2, Oct. 1999, pp. 548–554.
- [9] Wilkins, M. R., "From proteins to proteomes: large scale protein identification by two dimensional electrophoresis and amino acid analysis," *Biotechnology*, 1996, p. 5.
- [10] Giaever, G., et al., "Functional profiling of the *Saccharomyces cerevisiae* genome," *Nature*, Vol. 418, 25 July 2002, pp. 387–391.
- [11] Winzeler, E. A., et al., "Functional characterization of the *S. cerevisiae* genome by gene deletion and parallel analysis," *Science*, Vol. 285, 6 Aug. 1999, pp. 901–906.
- [12] Kim, S. K., et al., "A gene expression map for *Caenorhabditis elegans*," *Science*, Vol. 293, 14 Sept. 2001, pp. 2087–2092.
- [13] Kanehisa, M., and P. Bork, "Bioinformatics in the post-sequence era," *Nat. Genet.*, Vol. 33 Suppl., Mar. 2003, pp. 305–310.
- [14] Jones, S., and J. M. Thornton, "Principles of protein-protein interactions," *Proc. Natl. Acad. Sci. USA*, Vol. 93, 9 Jan. 1996, pp. 13–20.
- [15] Ito, T., et al., "Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins," *Proc. Natl. Acad. Sci. USA*, Vol. 97, 1 Feb. 2000, pp. 1143–1147.
- [16] Holford, M., et al., "VitaPad: visualization tools for the analysis of pathway data," *Bioinformatics*, Vol. 21, 15 Apr. 2005, pp. 1596–1602.
- [17] Nadkarni, P. M., et al., "Organization of heterogeneous scientific data using the EAV/CR representation," *J. Am. Med. Inform. Assoc.*, Vol. 6, Nov.-Dec. 1999, pp. 478–493.
- [18] Almaas, E., et al., "Global organizations of metabolic fluxes in the bacterium *Escherichia coli*," *Nature*, Vol. 427, 2004, pp. 839–843.
- [19] Barabasi, A. L., and R. Albert, "Emergence of scaling in random networks," *Science*, Vol. 286, 15 Oct. 1999, pp. 509–512.
- [20] Erdős, P., and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci.*, Vol. 5, 1960, pp. 17–61.
- [21] Vázquez, A., et al., "Modeling of protein interaction networks," *ComplexUs*, Vol. 1, 2003, pp. 38–44.
- [22] Jeong, H., et al., "Lethality and centrality in protein networks," *Nature*, Vol. 411, 2001, pp. 41–42.
- [23] Dorogovtsev, S. N., and J. F. F. Mendes, "Evolution of networks," *Adv. Phys.*, Vol. 51, 2002, pp. 1079–1187.
- [24] Li, L., et al., "Towards a theory of scale-free graphs: definition, properties, and implications," *Internet Math.*, 2005.
- [25] Balazsi, G., A. L. Barabasi, and Z. N. Oltvai, "Topological units of environmental signal processing in the transcriptional regulatory network of *Escherichia coli*," *Proc. Natl. Acad. Sci. USA*, Vol. 102, 31 May 2005, pp. 7841–7846.
- [26] Kashtan, N., et al., "Topological generalizations of network motifs," *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, Vol. 70, Sept. 2004.
- [27] Shen-Orr, S., et al., "Network motifs in the transcriptional regulation network of *Escherichia coli*," *Nat. Genet.*, Vol. 31, pp. 64–68.
- [28] Tong, A. H., et al., "A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules," *Science*, Vol. 295, 11 Jan. 2002, pp. 321–324.

- [29] Schwikowski, B., P. Uetz, and S. Fields, "A network of protein-protein interactions in yeast," *Nat. Biotechnol.*, Vol. 18, Dec. 2000, pp. 1257–1261.
- [30] Hartwell, L. H., et al., "From molecular to modular cell biology," *Nature*, Vol. 402, 2 Dec 1999, pp. C47–C52.
- [31] Yook, S. H., Z. N. Oltvai, and A. L. Barabasi, "Functional and topological characterization of protein interaction networks," *Proteomics*, Vol. 4, Apr. 2004, pp. 928–942.
- [32] Jeong, H., et al., "The large-scale organization of metabolic networks," *Nature*, Vol. 407, 5 Oct. 2000, pp. 651–654.
- [33] Ravasz, E., et al., "Hierarchical organization of modularity in metabolic networks," *Science*, Vol. 297, 30 Aug. 2002, pp. 1551–1555.
- [34] Girvan, M., and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. USA*, Vol. 99, 2002, pp. 7821–7826.
- [35] Gerdes, S. Y., et al., "Experimental determination and system level analysis of essential genes in *Escherichia coli* MG1655," *J. Bacteriol.*, Vol. 185, Oct. 2003, pp. 5673–5684.
- [36] Kobayashi, K., et al., "Essential *Bacillus subtilis* genes," *Proc. Natl. Acad. Sci. USA*, Vol. 100, 15 Apr. 2003, pp. 4678–4683.
- [37] Alterovitz, G., V. Muralidhar, and M. Ramoni, "Gene lethality detection across biological network domains: Hubs versus stochastic global topological analysis," in *IEEE GENSIPS 2006*, College Station, TX: IEEE, 2006.
- [38] Dykxhoorn, D. M., and J. Lieberman, "The silent revolution: RNA interference as basic biology, research tool, and therapeutic," *Annu. Rev. Med.*, Vol. 56, 2005, pp. 401–423.
- [39] Westphal, C. H., and P. Leder, "Transposon-generated 'knock-out' and 'knock-in' gene-targeting constructs for use in mice," *Curr. Biol.*, Vol. 7, 1 July 1997, pp. 530–533.
- [40] Alterovitz, G., V. Muralidhar, and M. F. Ramoni, "Gene lethality detection and characterization via topological analysis of regulatory networks," *IEEE Transactions on Circuits and Systems I*, Vol. 53, Nov. 2006, pp. 2438–2443.
- [41] Han, J.-D., et al., "Evidence for dynamically organized modularity in the yeast protein-protein interaction network," *Nature*, Vol. 430, 2004, pp. 88–93.
- [42] Ashburner, M., et al., "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium," *Nat. Genet.*, Vol. 25, May 2000, pp. 25–29.

Bayesian Networks for Genetic Analysis

Paola Sebastiani and María M. Abad-Grau

8.1 Introduction

The expansion of human genetic data of the past two decades presents the biomedical community with novel research opportunities but also novel computational challenges. The map of our chromosomes produced by the Human Genome Project documented similarities and differences between individuals and showed that the DNA is the same across individuals with the exception of about 0.1% nucleotide bases. Some of these variants may be the cause of monogenic diseases and, over the past decade, about 1,200 disease-causing genes have been identified through positional cloning. These successes have been mainly the result of phenotype-specific studies, focused on a single or a very limited number of observable characters, and based on linkage mapping of highly controlled individuals [1, 2].

However, most traits of medical relevance do not follow the rules of simple Mendelian monogenic inheritance and many common diseases such as diabetes, cardiovascular disease, and dementia, as well as longevity, are presumed to be determined by the joint action of several genotypes and their interaction with environmental factors [2]. These complex traits are characterized by high unpredictability, compared to traditional Mendelian disease in which a single mutation initiates a deterministic process leading to disease, and they can be compared to general complex systems, characterized by a high level of unpredictability of the output (the disease) given the input (genes and/or environmental exposure). The deciphering of these complex traits is one of the current overwhelming challenges of biomedical research.

As noted in the pioneering article of Eric Lander [2], the difficulty with the genetic dissection of these complex phenotypes is the stochastic nature of the association between genotypes and phenotypes that results in the same genotype leading to different phenotypes because of chance, effects of the environment, or interaction with other genes. It is acknowledged that positional cloning has limited power in the identification of variants leading to complex traits, and association studies collecting genotypes of several candidate genes hold the promise to cast light on the genetic basis of many complex diseases [1, 3].

This task is now made possible by the availability of high-throughput arrays for genome-wide genotyping that use the latest information from the HapMap project to tag the human genome with a sufficient number of variants [4]. The number of variants can be as large as 317,000 in the Sentrix HumanHap300 genotyping bead-chip (Illumina, San Diego, CA) and includes markers derived from the International HapMap Project (www.hapmap.org).

The remaining of this chapter is structured as follows: We review the necessary elements of population genetics in the next section. Section 8.3 introduces Bayesian networks as a tool to describe complex traits and Section 8.4 describes two applications. A discussion of open issues concludes in Section 8.5.

8.2 Elements of Population Genetics

The genetic basis of Mendelian diseases lies in the polymorphism of a single gene (a variation of the gene DNA sequence) that determines a nonsynonymous change in the amino acid produced during the gene expression, thus determining—directly or indirectly—the disease. In an organism that has two copies of each of its chromosomes (a diploid organism), an individual's genotype for a gene is determined by the pair of DNA coding of the same gene (alleles) occupying a given locus on a chromosome. The genotype can be coded either as the presence/absence of the minor allele (the allele with a smaller frequency in the population) in the two loci of the chromosome pair or as the complete allele pair that can be homozygous for the major allele (*wild type*), homozygous for the minor allele (*the mutant allele*), or heterozygous when the two alleles are different. The need for both mutant alleles characterizes a recessive disease, while the need of only one mutant allele determines a dominant disease. A well-known example of a recessive Mendelian disease is sickle cell anemia, caused by a single mutation of the β -globin gene, which determines a variant of the hemoglobin protein [5]. The common hemoglobin (wild allele) is a protein in red blood cells that carries oxygen to all parts of the body and gives blood its red color. The variant hemoglobin (mutant allele) in subjects with sickle cell anemia causes the red blood cells to become curved like a sickle. These sickle cells do not move through blood vessels as easily as round red blood cells, and tend to get stuck and possibly block the flow of blood to the limbs and organs. This can cause pain, organ damage, stroke, and serious anemia.

Contrary to classical Mendelian diseases that are caused by the polymorphism of a single gene, complex traits are those diseases thought to be determined by the co-occurrence of several genetic factors that by themselves would be unable to modulate disease [2]. Single nucleotide polymorphisms (SNPs)—variants of the DNA sequence that occur in at least 1% of the population—have become an invaluable resource to map complex traits in the human genome. SNPs are DNA sequence variations that occur when a single nucleotide (A, T, C, or G) in the genome sequence is altered. For example, an SNP might change the DNA sequence AAG-GCTAA to ATGGCTAA [6]. SNPs, as with other subtle variations of the genome across individuals, have been shown to be indeed one of the essential keys to understand how variations in the genome of different individuals correlate with observable traits of interest (phenotypes)—such as susceptibility to diseases or

response to treatment—and identify the genetic determinants of complex traits [7–9]. Although SNPs may determine important mutations in proteins and be causative of disease, often they merely act as flags on the genome to identify DNA regions that are associated with a phenotype. This is based on the extent of *linkage disequilibrium* (LD) of a DNA sequence: LD is defined by the nonrandom association of two or more loci and leads to the fact that segments of the genome are transmitted from parents to offsprings in blocks of limited or no recombination [10]. Therefore, SNPs that are associated with disease susceptibility and are in blocks of limited recombination can point to DNA regions implicated with the disease [4]. It has to be noted that LD can be observed between far apart loci or even different chromosomes that are not in linkage as an effect of gene interaction or nonadaptive processes as population structure, inbreeding, and stochastic effects. In other words, SNPs become markers of regions potentially associated with a trait, as shown in Figure 8.1.

In case-control studies, with the phenotype defined as the presence/absence of a trait, the data available for this discovery process are typically genotypes of case and control subjects at polymorphic loci, together with information about clinical covariates and environmental factors. The genotype can be coded as either the presence/absence of the minor allele (the allele with the smaller frequency in the population) in the two loci of the chromosome pair, or as the complete allele pair that can be homozygous for the major allele, homozygous for the minor allele, or heterozygous when the two alleles are different. Associations of the genetic variants with the phenotype of cases and controls are then examined using standard statistical techniques, namely tests of association in contingency tables or logistic regression, applied to each individual SNP.

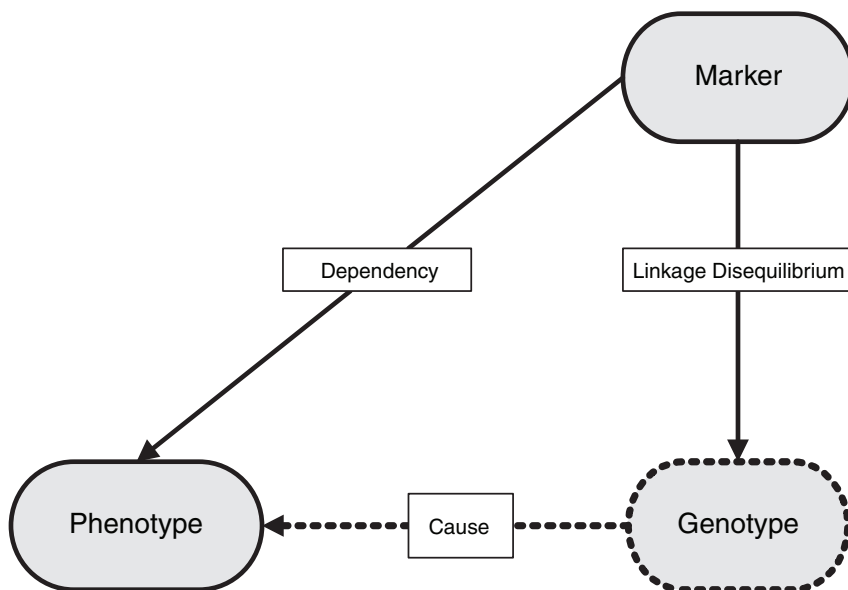


Figure 8.1 Representation of an SNP as marker by linkage disequilibrium. The association between phenotype and marker is the observable dependency that can be assessed in an association study. This association is induced by the linkage between the marker and the genotype that is causative of the disease.

Quantitative strategies for the discovery of genetic variants associated with human disease are still in their infancy, and it is widely acknowledged that the analysis of genotype/phenotype data produced by large genetic studies requires methodological developments and better solutions than those currently available to genetic epidemiologists [11, 12]. Reviews by Risch [13], Cardon & Bell [14], and Hoh & Ott [15] survey the major approaches to the statistical analyses of multiple genotypes in either case control or family-based studies and highlight the limitations of the one-snp-at-a-time procedure that is predominantly used to analyze these data. This procedure examines one genetic variant at a time in relation to a presumably well-defined phenotype using standard association tests [16]. This reductionistic approach risks not capturing the multigenic nature of complex traits and typically identifies too many associations because of dependencies between SNPs in contiguous regions as a result of LD and the evolutionarily induced dependency between SNPs on different chromosomes [17]. A further limitation of the one-at-a-time variant approach is the inability to discover associations that are due to interdependent multiple genotypes. Hoh & Ott [15] point out the case in which the simultaneous presence of three genotypes at different loci leads to a disease. The three genotypes themselves have the same marginal penetrance and would not be found associated with the disease in a one-at-a-time search. This situation is an example of the well-known Simpson's paradox [18] and the fact that marginal independence of two variables does not necessarily imply their conditional independence when other variables are taken into account [19].

These two situations are described in Figure 8.2, which shows a possible dependency structure between a phenotypic trait (node P) and four SNPs (nodes G1—G4 in the graph). The two SNPs G2 and G3 are associated with the trait. The SNP G1 is an older SNP that is associated with the SNP G2 through evolution. The two SNPs do not need to be necessarily on the same genes, or on the same region of linkage disequilibrium. The SNP G4 is not associated with the phenotype individually, but only in conjunction with SNP G3. A simple one-at-a-time search would probably lead to identify G1 as associated with P and hence introduce redundancy, if G1 and G2 are in linkage disequilibrium, or introduce a false association if G1 and G2 are not on the same gene or region of linkage disequilibrium. Another risk would be to lose the association of G4 with P.

Multivariate statistical models can circumvent these limitations by examining the overall dependency structure between genotypes, phenotype, and environmental variables. A typical solution is to resort to multivariate logistic regression models to describe the odds for the disease, given particular genotypes. Logistic regression models can be used to assess whether the association between phenotype and genotypes is confounded by external factors such as population admixture [20] and whether an external factor or an environmental exposure is an effect the modifier of an association [16]. However, they pose three serious limitations:

1. When the susceptibility to disease is caused by the interaction among several genes, the number of parameters required to fit a logistic regression model increases at an exponential rate.
2. The genotypes are treated as covariates rather than random variables so that genotyping errors and missing genotypes are difficult to handle.

3. Logistic regression can examine the association between one well-defined phenotypic character at a time.

Overcoming these limitations requires novel techniques that go beyond “traditional statistical thinking” in order to accommodate the complexity of genetic models. Knowledge discovery techniques developed by the machine learning community in the last decades have the potential to help the discovery of the genetic basis of complex traits [21]. Partitioning techniques [22], methods for dimensionality reduction [23, 24], and tree-based association analysis have already been proposed to analyze large association-based and family-based studies [25, 26]. Progresses in Bayesian computations are also making Bayesian techniques available to the genetic community [27].

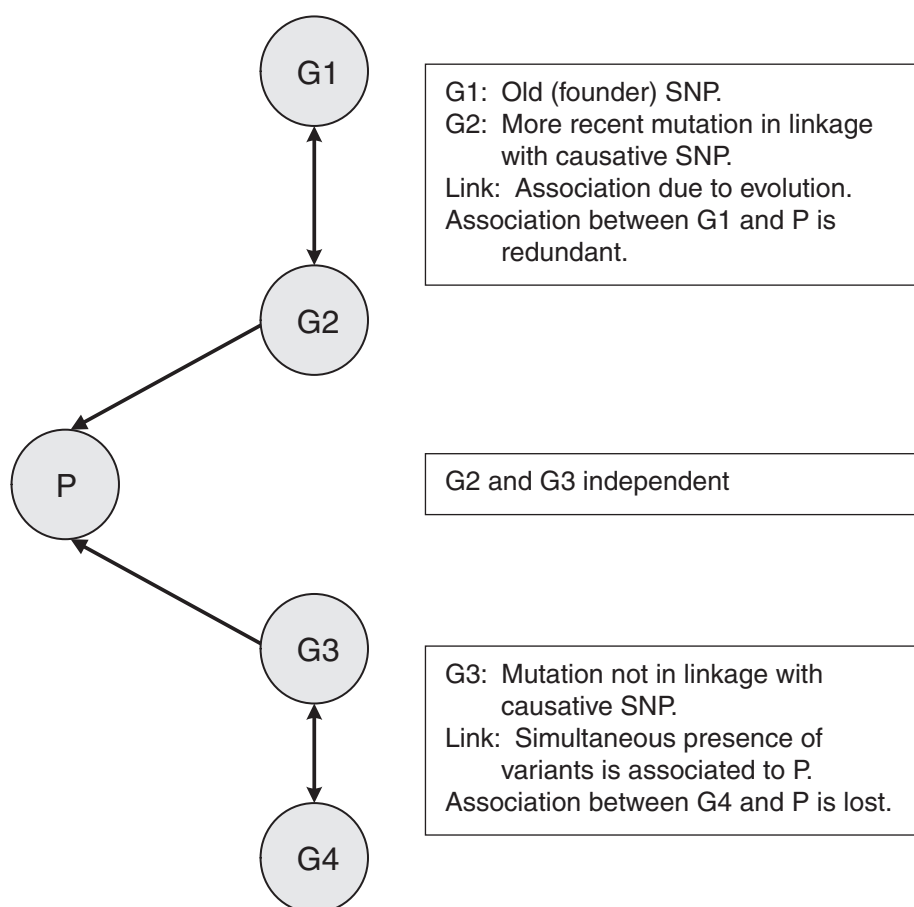


Figure 8.2 A simple dependency structure between a phenotypic trait (node P) and four SNPs (nodes G1–G4 in the graph). SNPs G2 and G3 are associated with the trait. SNP G1 is an older SNP that is associated with SNP G2 through evolution. SNP G4 is not associated with the phenotype individually, but only in conjunction with SNP G3. A simple one-at-a-time search would lead to identify G1 as associated with P, and hence introduce redundancy, and may lose the association of G4 with P.

To conclude this section, we note that the term *complex trait* is also used to describe phenotypes that may not be considered as a single, well-defined disease but are determined by multiple variables contributing to the disease condition or syndrome (e.g., metabolic syndrome). An example is exceptional longevity (EL), which can be defined in numerous ways if secular trends in life expectancy, gender, and social, environmental, and behavioral factors are taken into account [28, 29]. Definitions of EL may include survival past a specified extreme age and/or disability-free survival past a specified age and may be the result of a lack of genetic variations that predispose to age-related diseases and premature mortality (particularly cardiovascular disease, cancer, and stroke), but it could also be attributed to genetic variations that are protective against aging and might delay the onset of age-related diseases [30, 31]. The search for the genetic basis of EL faces two levels of complexity: the definition of the phenotype as determined by multiple factors as well as the search for a possibly complex network of genes that interact to modulate longevity.

8.3 Bayesian Networks

Bayesian networks (BNs) [19, 32] are models able to approximate complex and uncertain systems by stochastic processes. These network models offer the advantage of modularity in order to build a complex system by building linkable modules. Modularity provides the critical advantage of breaking down the discovery process into the specific components of a complex model, thus simplifying the processes of model discovery from the data, as well as the interpretation of the findings represented by the model, and its usage for prediction and explanation.

8.3.1 Representation

A BN is a multivariate dependency model in which the joint probability distribution of a set of random variables, $X = (X_1, X_2, \dots, X_n)$, factorizes according to the marginal and conditional independencies described by a directed acyclic graph (DAG). Nodes in the graph represent the random variables, and directed arcs from parents to a child node define directed stochastic dependencies quantified by probability conditional distributions.

Figure 8.3 depicts three BNs. Figure 8.3(a) is a simple network describing the dependency of a phenotypic character P on a single SNP, G . Both the phenotype and the SNP are treated as random variables, with a joint probability distribution. The directed graph represents a decomposition of the joint probability distribution of the two variables according to the flows of the arcs in the graph: the joint probability distribution factorizes into the product of the marginal distribution of G —the *parent* node—and the conditional distribution of P —the *child* node—given G . As an example, to compute the joint probability for $G = AA$ and $P = \text{“Absent”}$ we use the factorization

$$p(G = AA, P = \text{Absent}) = p(G = AA) p(P = \text{Absent} | G = AA) = 0.6 \times 0.3 = 0.18$$

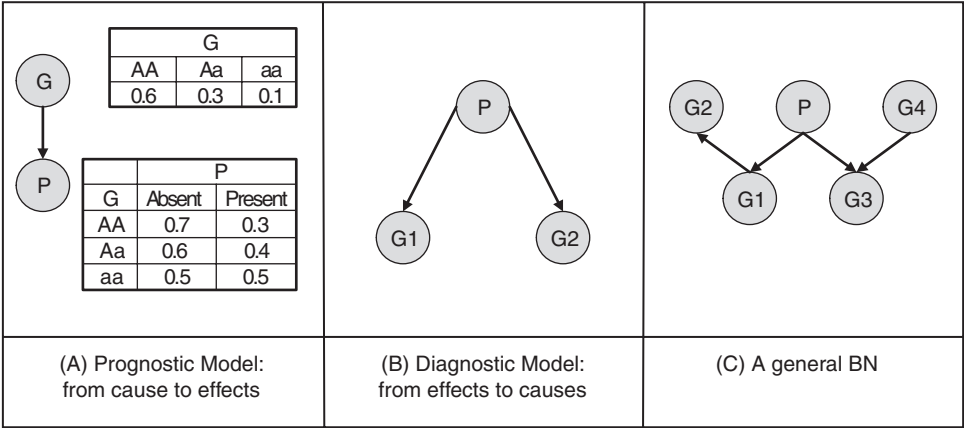


Figure 8.3 Examples of BN structures. (a) A simple BN with two nodes representing a SNP (G) and a phenotype (P). The probability distribution of G represents the genotype distribution in the population, while the conditional probability distribution of P describes the distribution of the phenotype given each genotype. (b) The association between SNPs and phenotype can be reversed using Bayes theorem. (c) A BN linking four SNPs (G1–G4) to a phenotype P. The phenotype is independent of G2, once we know SNPs G1, G3, and G4. The joint probability distribution of the network is fully specified by the five distributions representing the distribution of G1 given P, of G2 given G1, of G3 given P and G4 (six parameters), of G4, and of P. By Bayes theorem, one can compute the probability distribution of the phenotype, given a genetic profile specified by the SNPs G1, G3, and G4 that are the sufficient predictors of the phenotypes.

The marginal and conditional probability distributions are sufficient to define the association between *P* and *G* because their product determines their joint probability distribution. This property persists when the direction of the arc is inverted in the graph (Figure 8.3b), in which the arcs are directed from the phenotype to the genotypes of two SNPs, and when the graphical structure is expanded to include several variables (Figure 8.3c); the overall association is measured by the joint distribution that is still defined by the product of each child-parent conditional distribution. This modular nature of a BN is due to the conditional and marginal independencies among the variables encoded by the directed acyclic graph.

Conditional and marginal independence are substantially different concepts. For example, two variables can be marginally independent, but they may be dependent when we condition on a third variable. The directed acyclic graph in Figure 8.3(c) shows this property: the two nodes *P* and *G4* are marginally independent, but they become dependent when we condition on their common child: the node *G3*. A well-known consequence of this fact is Simpson’s paradox [18], and a typical application in genetics is the dependency structure of genotypes among members of the same family: the genotypes of two parents are independent, assuming random mating, but they become dependent once the genotype of their common child is known.

Conversely, two variables that are marginally dependent may be made conditionally independent by introducing a third variable. This situation is represented by the directed acyclic graph in Figure 8.3(b), which shows two children nodes (*G1* and *G2*) with a common parent *P*. In this case, the graph shows that the two children nodes are independent, given the common parent, but they may become de-

pendent when we marginalize out the common parent. Suppose, for example, the three variables represent the presence/absence of an X-linked genetic marker in the mother genotype (P) and the children genotype ($G1$ and $G2$). The marginal distribution of P represents the prevalence of the marker in the population, and the conditional probabilities associated with the nodes $G1$ and $G2$ represent the probability that each child has the marker, given the maternal genotype.

These examples are special cases of marginal and conditional independences, and a directed acyclic graph can actually represent a variety of different situations [33, 34]. The overall set of marginal and conditional independencies represented by a directed acyclic graph is summarized by the local and global Markov properties.

The *local Markov property* states that each node is independent of its non-descendant, given the parent nodes, and leads to a direct factorization of the joint probability distribution of the network variables into the product of the conditional distribution of each variable, given its parents. This factorization provides modules defined by each variable and the set of its parents as the sufficient components to characterize the overall probability distribution. As an example, G is the parent node of P in Figure 8.3(a) and the factorization of the joint probability distribution into the marginal distribution of G and the conditional distribution of P , given G , is a simple application of the local Markov property. Similarly, the conditional independence of $G1$ and $G2$, given P , in Figure 8.3(b) is again an application of the local Markov property because P is the parent node of $G1$ and $G2$ and separates them from each other. More formally, the local Markov property provides the following factorization of the joint probability distribution

$$p(x_{1k}, x_{2k}, \dots, x_{vk}) = \prod_{i=1}^v p(x_{ik} | \pi_{ij})$$

Here, $x_k = (x_{1k}, x_{2k}, \dots, x_{vk})$ is a combination of values of the variables in X . For each i , the variable Π_i denotes the parents of the variable X_i , while x_{ik} and π_{ij} denote the events $X_i = x_{ik}$ and Π_i . Particularly, π_{ij} is the combination of values of the parent variable Π_i in $X = (x_{1k}, x_{2k}, \dots, x_{vk})$.

The *global Markov property*, on the other hand, summarizes all conditional independencies embedded in the directed acyclic graph by identifying the *Markov blanket* of each node. This is defined as the set of parents and children nodes, as well as the parents of the children nodes. For example, the Markov blanket of the node P in Figure 8.3(c) is the set of nodes $G1$ and $G3$ (the children nodes) and the node $G4$ (parent of child node $G3$). Knowing the states of these three nodes, therefore, is sufficient to infer the distribution of P . The global Markov property is the foundation of many algorithms for probabilistic reasoning with Bayesian networks that allows for the investigation of undirected relationships between the variables, and their use for making prediction and explanation [33].

The modular representation of a BN is able to capture complex dependency models that integrate associations between SNPs and phenotype, associations between SNPs due to LD or evolution, and interaction processes linking SNPs, phenotype, and environmental factors with a small number of parameters [35]. The network in panel Figure 8.3(c) is an example of structure that captures the associations of two SNPs, $G1$ and $G3$, with the phenotype. The association of SNP $G4$

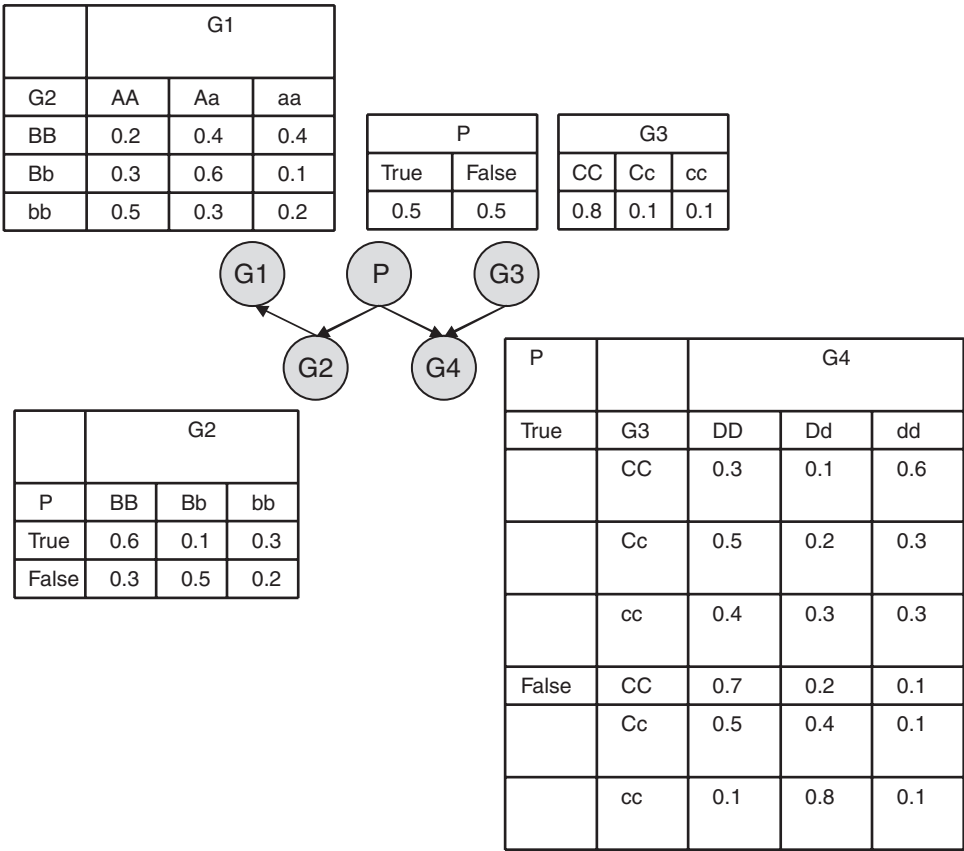


Figure 8.4 Conditional probability tables that are necessary to quantify the network in Figure 8.3(c). The local Markov property ensures that these conditional probability tables specify the full joint distribution, consistent with the set of marginal and conditional independencies represented by the directed acyclic graph.

appears conditional only on SNP G3, and the SNP G2 is redundant, given G1. This network describes the structure of association outlined in Figure 8.2 through marginal and conditional independence. Furthermore, the local Markov property ensures that the full joint probability distribution of the variables in the network is completely specified by the conditional probability distributions of each variable, given its parents. See Figure 8.4 for an example.

8.3.2 Learning

The modularity of BNs allows us to define large dependency networks from comparatively small datasets, and well-established techniques exist to induce BNs from data in an almost automated manner [19, 36]. Strategies for learning BNs from data include standard association tests, such as those implemented in Tetrad II, or Bayesian model search procedure. Our experience with Bayesian procedures is that they are usually more robust to false associations. This is due to the use of proper prior distributions that can tune the level of evidence needed to accept an association.

The main intuition of a Bayesian model selection strategy is to assess each dependency model M_b by using as scoring metric the model posterior probability [37]

$$p(M_b | D) = \frac{p(D | M_b)p(M_b)}{p(M_b)}$$

The quantity $p(M_b)$ is the prior probability of the model M_b before seeing the data D . The quantity $p(D | M_b)$ is the *marginal likelihood*, and it is computed as follows. Given the Bayesian network M_b , let θ^b denote the vector parameterizing the joint distribution of the variables $X = (X_1, X_2, \Lambda, X_v)$. In other words, the vector θ^b describes the marginal and conditional probabilities (or parameters of the marginal and conditional distributions) that quantify the network. We denote by $p(\theta^b)$ the prior density of θ^b . The likelihood function is $p(D | \theta^b)$, and the marginal likelihood is computed by averaging out θ^b from the likelihood function $p(D | \theta^b)$. Hence

$$p(D | M_b) = \int p(D | \theta^b) p(\theta^b) d\theta^b$$

The computation of the marginal likelihood requires the specification of a parameterization of each model M_b , and the elicitation of a prior density for θ^b .

When the variables $X = (X_1, X_2, \Lambda, X_v)$ are all discrete, the parameter vector θ^b consists of the conditional probabilities $\theta_{ijk}^b = p(X_i = x_{ik} | \Pi_i = \pi_{ij}, \theta)$. In this case, it is easy to show that, under the assumption of multinomial sampling with complete data, the likelihood function becomes

$$p(D | \theta^b) = \prod_{ijk} \theta_{ijk}^{n_{ijk}}$$

where n_{ijk} is the sample frequency of pairs (x_{ik}, π_{ij}) in the database D . A convenient choice of prior for the parameter vector θ^b is the Hyper-Dirichlet distribution. This is defined as a set of independent Dirichlet distributions $D(\alpha_{ij1}, \Lambda, \alpha_{ijci})$, one for each set of parameters $\{\theta_{ijk}^b\}_k$ associated with the conditional distribution $X_i | \pi_{ij}$. It is well known (see [19]), that this choice for the prior distribution provides the following formula for the marginal likelihood:

$$p(D | M_b) = \prod_{ijk} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})}$$

Here, $n_{ij} = \sum_k n_{ijk}$ is the marginal frequency of π_{ij} in the database, and $\alpha_{ij} = \sum_k \alpha_{ijk}$.

For consistent model comparisons, it is convenient to adopt symmetric Hyper-Dirichlet distributions, which depend on one hyperparameter α , called *global precision*. Each hyperparameter α_{ijk} is computed from α as $\alpha_{ijk} = \alpha / (q_i c_i)$, where c_i is the number of categories of the variable X_i , and q_i is the number of categories of the parent variable Π_i . The rationale behind this choice is to distribute the overall prior precision α in a uniform way among the parameters associated with different conditional probability tables. In this way, the prior probabilities quantifying each network are uniform, and all the prior marginal distribution of the network variables are uniform and have the same prior precision.

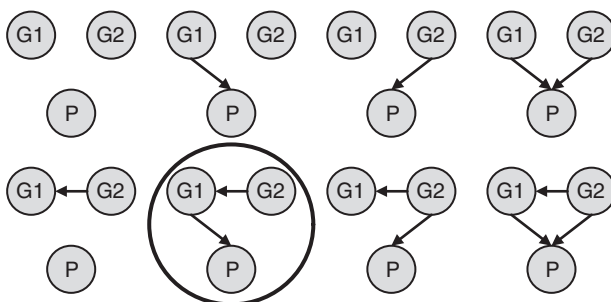
The use of the model posterior probability as a scoring metric leads to select the model that is most likely, given the evidence provided by the data. An important advantage of using BNs in genetic analysis is their modularity, induced by the local Markov property. Under regularity assumptions, the marginal and conditional independencies induce a factorization of the posterior probability of each model M_b into factors proportional to the posterior probability of the dependency of each node on its parent nodes

$$p(M_b|D) = \prod_{i=1}^v p(M_b^i|D) \propto \prod_{i=1}^v p(D|M_b^i) p(M_b^i)$$

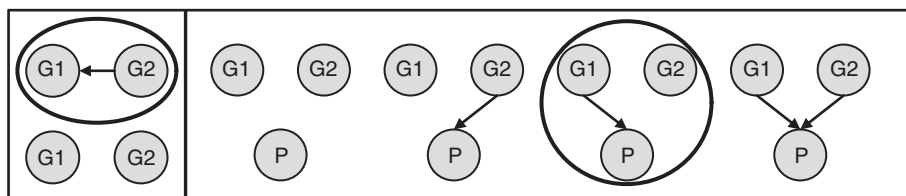
By this modularity, we can build a complex network by building local models M_b^i for each variable X_i , and these local models can then be joined together as in standard path analysis [35].

This strategy is exemplified in Figure 8.5 and requires an initial ordering of the variables to avoid the introduction of loops. For example, suppose that we are looking for a prognostic model, possibly relating the genotypes of two SNPs, G1 and G2 to a phenotypic trait P. Suppose also that G2 is an older SNP, so that we can limit attention to the dependency of G1 on G2 rather than the other way around, to represent linkage due to evolution. These constraints would limit the set of possible models to those eight models represented in the top of Figure 8.5, and a full search

Full Search:



Modular Search:



Join by Common
Nodes:



Figure 8.5 Computational gain of a modular search, compared to a full search. The full search would require computing the Bayesian score for each of the eight models describing different dependency structures between two SNPs and a phenotype. The model selected by this search is circled in the top panel, and it can be described as the union of the two models circled in the middle panel, as shown in the bottom panel.

would require the computation of the Bayes score for each model so that to select the model that scores best (the one circled in the figure).

The idea for the modular search is based on the observation that each of the eight models can actually be decomposed into two simpler models: one describing the dependency of P on G1 and G2, and the other one describing the relation between the two SNPs, G1 and G2. The best model can then be searched by finding the best model for P, and the best model for G1, and then linking them through the variables in common.

This modularity is critical when searching for a large model of dependency with many variables. In the particular case of complex traits, the *common disease, common allele variants* assumption [38] would support the fact that the genetic basis of common disease is given by the simultaneous presence of many alleles that are common in the population. Individually, each allele would not be sufficient to predict the disease outcome, which can be predicted only by their simultaneous presence. Therefore, in deciphering the genetic basis of complex traits, we expect to see a large number of SNPs involved in modulating disease severity, each having a small effect on the phenotype.

Another “trick” that we have found useful to induce networks from large datasets of variables is to use diagnostic rather than prognostic models, in which we model the dependency of SNPs on the phenotype [see panels (b) and (c) in Figure 8.3]. This structure has the advantage of representing the correct data-generating mechanism of a cross-sectional study rather than the physical/causal process underlying the biological system that relates genotype to phenotype. The prediction of the phenotype, given a particular genetic profile, is not explicitly represented by this model structure, but needs to be computed by using Bayes theorem [19]. There are standard probabilistic algorithms for these calculations that are implemented in software for BN modeling and reasoning such as Bayesware Discoverer [35].

A particular advantage of this “inverted” dependency structure is the ability to represent the association of independent as well as interacting SNPs with the phenotype. Furthermore, this structure is able to capture more complex models of dependency [39] compared with regression models because the association of each SNP with the phenotype does not affect the association of other SNPs with the phenotype. In contrast, in regression structures the presence of an association between a SNP and the phenotype affects the association between the phenotype and other SNPs, thus imposing a limit on the number of SNPs that can be detected as associated with the phenotype.

Crucial to the feasibility of the learning process is the use of efficient and accurate model search strategies when the space of possible models makes even the modular search unfeasible. One of the most popular and efficient search algorithms to induce BNs from data is a greedy search algorithm known as the K2 [40]. This search algorithm explores a space of different dependency models that are consistent with an initial ordering of the variables, scores each model by its probability conditional on the available data, and returns the model with maximum posterior probability in the set of explored models. Efficiency can be gained by ordering the SNPs according to their entropy so that more entropic SNPs could be dependent only on less entropic SNPs. This heuristic is used, for example, to build standard classification and regression trees, and we have used it successfully to build the net-

work describing the genetic dissection of stroke in sickle cell anemia subjects [41]. Improvements to the K2 strategy can be obtained by a stepwise search, or a stochastic search.

Last but not least is the ability to compute the Bayesian score in closed form. There are standard situations in which this is possible: for example, we have shown that when all the network variables are categorical, and the data do not have missing entries, the posterior probability of each network model can be computed in closed form. Another simple situation is when all the network variables are continuous, follow Gaussian distributions, and the relations are all linear [42]. In the context of genetic analysis this last situation does not apply because genotypes are categorical variables with two or three states. In these situations, it seems convenient to categorize continuous variables by using some forms of discretization. Besides the gain in computation, this procedure allows us to model nonlinear dependencies between variables in an automated way.

There are several discretization algorithms that could be used as a preprocessing step. Depending on the use of the variable to be classified, discretization methods can be divided into two groups [43]: *unsupervised* and *supervised* methods. Unsupervised discretization algorithms do not make use of other variables in the discretization process and comprise methods such as “equal width interval binning” or “equal frequency interval binning.” The former divides a continuous variable into k equally sized bins, where k is a parameter supplied by the user. The latter divides a continuous variable into k bins, where each bin contains n/k adjacent values, with n being the size of the sample. For prognostic or diagnostic models describing the association between a phenotype with genotypes and other clinical or environmental variables, the phenotype can be used to aid the discretization that becomes supervised. This is strongly recommended because supervised discretization usually outperforms unsupervised methods [43, 44]. Those methods using a stopping criterion that trades off accuracy with complexity seem to outperform others, as they can prevent overfitting and improve accuracy. An example is the discretization algorithm based on the *Minimum Description Length* (MDL) induction principle proposed by Fayyad and Irani [45].

Once the network structure is induced from data, the last task is to quantify the conditional distributions that are sufficient to fully specify the joint distribution. In the context of categorical variables, this last step can be efficiently performed by standard Bayesian conjugate analysis. The idea is to essentially estimate conditional probabilities as relative frequencies in relevant samples, and smooth them by parameters that encode prior knowledge. See [35] and [41] for details.

8.3.3 Reasoning

One of the most important features of BNs is the fact that they can be used for a variety of probabilistic reasoning. This is possibly the feature that makes this model formalism the most flexible and useful in the set of models commonly used in statistics and machine learning. Regardless of the structure represented by the directed acyclic graph, the network can be used for prognostic reasoning (from cause to effects) as well as diagnostic reasoning (from effect to causes). Furthermore, inference does not need to be limited to one specific “response variable” as in standard

regression models: every node X in the network that represents a stochastic variable can be treated as “response variable” as well as a “regressor.” In the former case, we could be interested in computing the probability that X is in a particular state, given what we know about any set of variables in the network. In the latter case, the variable X could be in a known state and we might be interested in its effect on some other network variables.

Key to this probabilistic reasoning is Bayes’ theorem, which can be used to compute the posterior probability of a variable X , given a known state for a variable Y

$$p(X|Y) = \frac{p(Y|X)p(X)}{p(Y)}$$

In the context of BNs used for genetics, for example, X may represent a phenotypic trait and Y a particular SNP. Knowing the state of Y would be equivalent to knowing the genotype, and the conditional probability $p(P|G)$ would be the risk for the phenotype, given a genetic profile. The conditional probability that appears in the numerator of Bayes’ theorem—the quantity $p(G|P)$ —would be likely if data are collected in a case control study. Alternatively, X may represent a particular SNP and Y the phenotype, so that the conditional probability $p(G|P)$ would be the population frequency of a particular genotype, given a known disease status.

With complex network structures, the application of Bayes’ theorem requires sophisticated probabilistic reasonings that alternate marginalization steps with conditioning steps. Several algorithms are available, including stochastic algorithms such as Gibbs Sampling for approximate inference in very complex networks; see [19] and [34] for details. As an example, Figure 8.6 reports the conditional probability of the phenotype, given a particular genetic profile, for the network described in Figure 8.4. Note that in the computation of the conditional probability, we need only to know the status of the variables in the Markov blanket of the node P . However, knowledge of the Markov blanket is not necessary for the computation of the conditional probability of the phenotype. Bayes’ theorem can also be used to compute the risk for the phenotype given a partially known genetic profile: for ex-

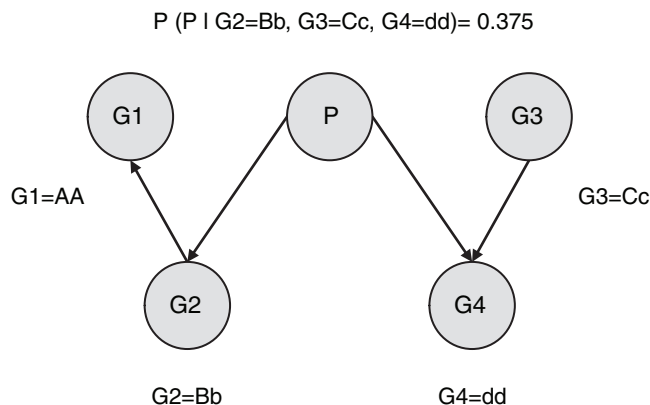


Figure 8.6 Risk for the disease, given a particular genetic profile represented by a combination of genotypes of four SNPs.

ample, the risk for the disease is $p(P|G2 = Bb) = 0.167$ if only the genotype $G2$ is known.

8.3.4 Validation and Inference

Because of the need to use a heuristic search in learning a BN from a set of variables, validation of the model is of critical importance. The selected model can be validated to assess its goodness of fit and predictive value. Residual analysis is the standard approach to examine the goodness of fit and in [35] we define blanket residuals as quantities to be used in residual analysis. The real validation, however, is the assessment of the predictive accuracy of the models. This can be done by intrinsic and extrinsic cross-validation techniques [39]. These are standard techniques to assess the accuracy of the network model in forecasting the phenotypes of independent individuals, given their genotypes. The original dataset is partitioned into k nonoverlapping subsets that are used repeatedly for learning the network of dependency and assessing its predictive accuracy. At each step, a training set of $k - 1$ subsets is used to induce the network, that is then tested on the last subset, called the test set. The assessment consists of predicting the phenotype of the subjects in the test set, given their genetic profile, and the accuracy is measured by the frequency of individuals for whom the correct phenotypes are predicted with probability larger than 0.5. With intrinsic cross-validation, only the conditional distributions that quantify the network are estimated from the training data, without searching for the best dependency structure. This test therefore assesses the robustness of the selected network to sampling variability. In extrinsic cross-validation, the best network structure is searched in each training set so that this method assesses the robustness of the associations to sampling variability.

Although cross-validation is often the best validation possible, the crucial validation remains the assessment of the best network in independent populations. Having such data available makes two types of validations possible: a predictive validation, in which the network is used to predict the phenotypes of the subjects in the independent set, given their genetic profile, and a model validation, in which a new network is induced from the independent data, using the same set of constraints imposed on the model space to induce the best network in the original study. The former validation assesses the accuracy of a model to predict the correct phenotypes. However, accuracy could be high even though there are some redundant or spurious associations in the network. The latter validation examines the reproducibility of the associations between genotypes and phenotypes to see whether these associations generalize across different populations [14, 46].

8.3.5 Risk Prediction

When we know in advance that our only interest is to obtain a predictive model, that is, there is one well-defined phenotype (the *class*) and we want to predict its values for any given configuration of the other variables (the *input variables*) there are several learning algorithms that can be used to build the prediction rule, also termed a *classifier*. A BN becomes a classifier whenever the graphical structure is not a DAG but a less complex graph. The simplest learning algorithm, called the

Naïve Bayes classifier, assumes that the input variables are conditionally independent, given the class. Thus the structure is always the same: for a set of n input variables, there are only n arcs in the graph from the class node to each variable. An example is the diagnostic model shown in Figure 8.3(b).

Other algorithms consider more complex graphical structures and perform first a search through the space of possible models. An example, known as a Tree Augmented Network (TAN) [47], is shown in Figure 8.7(a). In a TAN the class variable has no parents and each attribute has as parents the class variable and at most one more variable. In Figure 8.7(b) an example of a more general classifier is shown. This general structure is referred as Augmented Network (AN), and it allows any number of parents for each input attribute, considering that directed cycles are not allowed.

The predictive accuracy of very simple classifiers is strongly influenced by redundant variables, and accuracy may decrease exponentially [48]. There are several algorithms that perform a selection of the variables to be included in the classifier. In the context of risk prediction based on a genetic profile, redundant SNPs could be those SNPs in strong LD, for example. As far as we know, no specific algorithms tailored to genetic data have been proposed.

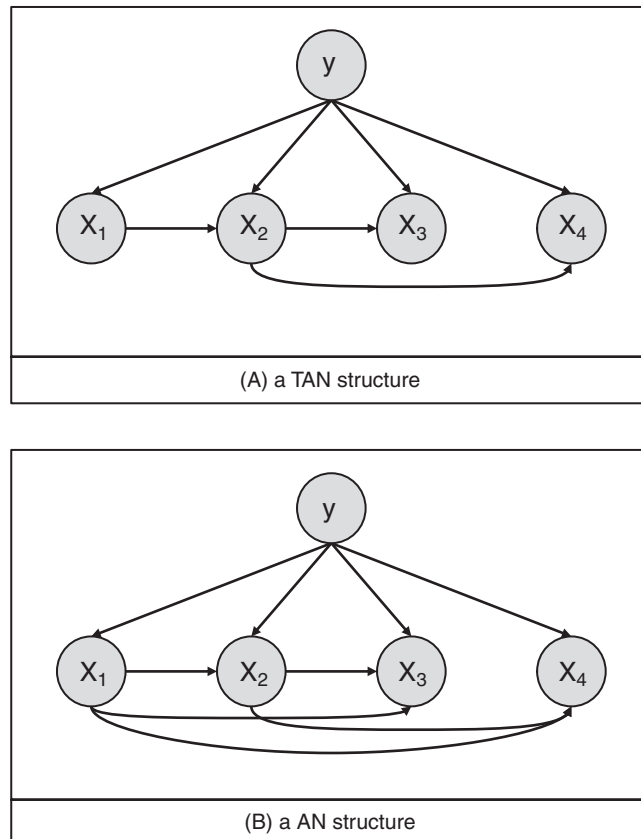


Figure 8.7 Two examples of Bayesian classifiers: (a) a TAN structure and (b) a general AN structure.

8.4 Two Applications

In this section we present two applications of BNs to describe the genetic basis of a complex phenotype and to describe an example of genetic model of a complex trait that is determined by multiple variables contributing to a disease condition, and by multiple genes.

8.4.1 Stroke Risk in Sickle Cell Anemia Subjects

The BN model developed [41] for the genetic dissection of stroke in subjects with sickle cell anemia is a small-scale example of the successful and powerful use of Bayesian networks to model the genetic basis of complex traits. This model was based on a BN that captures the interplay between genetic and clinical variables to modulate the risk for stroke in sickle cell anemia subjects. Available for this study were about 250 SNPs in candidate genes of different functional classes in 92 patients with nonhemorrhagic stroke and 1,306 in disease-free controls. The model was inferred from data using the Bayesian model search strategy described earlier, and the space of possible models was restricted to diagnostic models, so that the phenotype node was tested as associated with each SNPs. To further reduce the complexity of the search, SNPs were ordered by entropy so that less variable SNPs were tested only as parents of more variable SNPs. This search favors more recent mutations as possibly implicated with the phenotype.

The network identified by this search describes the interplay between 31 SNPs in 12 genes that, together with fetal hemoglobin (a protein that is present in adult subjects with sickle cell anemia), modulate the risk for stroke. This network of interactions included 3 genes, BMP6, TGFBR2, TGFBR3, with a functional role in the TGF- β pathway and also SELP. These genes and klotho (KL), a longevity-associated gene in animals and humans [49], are also associated with stroke in the general population. The model was validated in a different population of 114 subjects, including 7 stroke cases and 107 disease-free subjects, and reached a 100% true positive rate, 98.14% true negative rate, and an overall predictive accuracy of 98.2% [41]. The main features of this model are that it can be used as a diagnostic tool to identify those genetic profiles that increase the risk for stroke and hence increase knowledge about genetic modulators of stroke, as well as a prognostic model for risk prediction based upon different clinical presentations.

8.4.2 Network Representation of a Complex Trait

Dyslipidemia is a disorder of lipoprotein metabolism that may include overproduction or deficiency of lipoprotein. Dyslipidemia may be manifested by elevation of the total cholesterol, elevation of the “bad” low-density lipoprotein (LDL) cholesterol and the triglyceride concentrations, and a decrease in the “good” high-density lipoprotein (HDL) cholesterol concentration in the blood. By definition, this disease is determined by the abnormal condition of one or more variables. Our current knowledge supports the conjecture that nutrients modulate the expression of numerous gene products and that the relations between nutrients and gene products may change according to the overall genetic background [50]. Several genes have

already been implicated in dyslipidemia, including apolipoproteins A1, C3, A4, and A5 (APOA1, APOC3, APOA4, and APOA5), hepatic and lipoprotein lipase (LIPC and LPL), and cholestery ester transfer protein (CEPT) [51].

We used data from about 3500 subjects in the Framingham Heart Study to relate the genotypes of 24 SNPs in 14 genes and 18 variables that describe dyslipidemia. Because most of these variables take real values, we categorized them using quartiles and induced the most likely network of dependency (displayed in Figure 8.8) by using the Bayesian model search strategy described in the previous section. The network displays expected associations such as those between total cholesterol level (node CHOL), LDL-C and HDL-C (nodes LDL and HDL), and total triglyceride (node TG), or those between beta blockers (node BETA), BMI, age, and hypertension (node HTN). The associations between markers on the genes CEPT, LPL, and APOA are consistent with findings of other investigators. The novelty of this approach is, however, in the ability to model simultaneously several variables that characterize dyslipidemia, without focusing on the relations between individual measurements and individual genes.

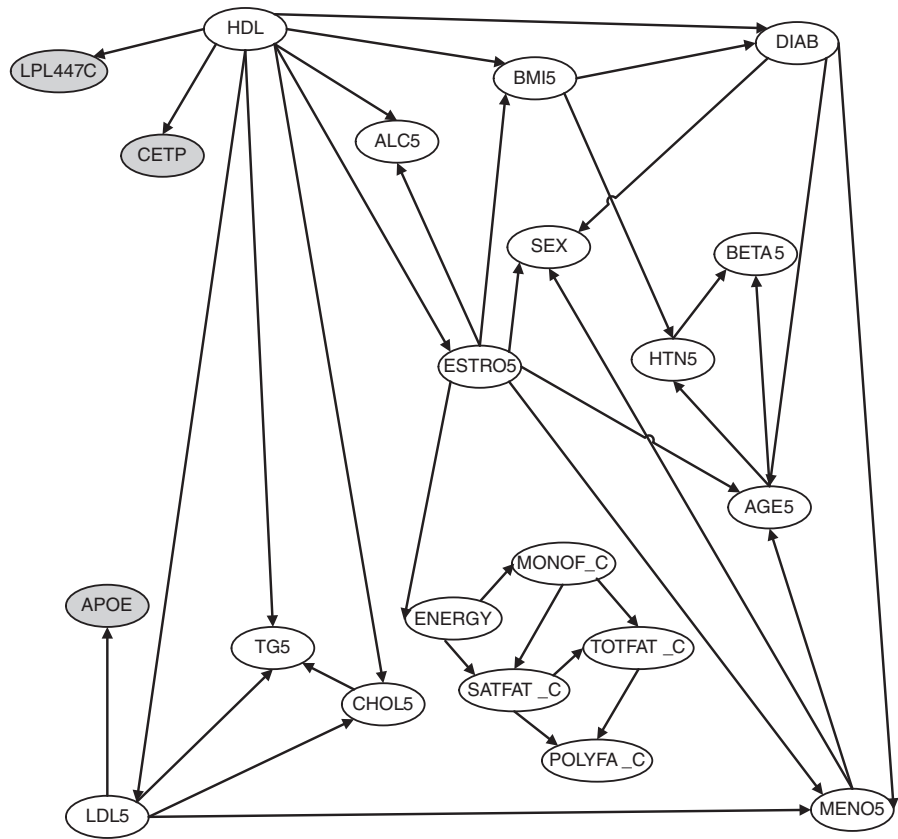


Figure 8.8 The BN displaying the structure of dependency between 18 variables and the genotypes of markers of three genes: LPL, CEPT, and APOE. Nodes in gray are markers on the genes CEPT, LPL, and APOE. The number 5 appended to some of the variables signify that those variables were measured at visit 5, approximately 10 years after the beginning of the study.

The discovery of somewhat expected associations provides evidence that the search strategy described earlier is able to discover true dependencies in the data. Furthermore, the associations between SNPs on CETP and LPL447, alcohol consumption (node ALC), and estrogen replacement therapy (node ESTRO) with HDL show how genetic factors modulate response to fat intake together with environmental and other factors. These and other associations in the network suggest that a complex model of gene-environment interactions modulates the risk for dyslipidemia.

This network was built using the program Bayesware Discoverer (<http://www.bayesware.com/>), a computer program for learning BNs using the Bayesian model selection strategy described earlier. The search algorithm in Discoverer explores a space of different dependency models that are consistent with a preordering of the variables, it scores each model by its probability conditional on the available data, and returns the model with maximum posterior probability.

The program provides details of the search strategy, and the strength of the associations selected as a result of the search procedure can be assessed by the Bayes factor that measures the odds of a model M_i versus a model M_j by the ratio of their posterior probabilities $p(M_i | \text{data}) / p(M_j | \text{data})$ [37]. A Bayes factor of magnitude b implies that the model M_i is b times more likely than the model M_j or, equivalently, that the posterior probability of the model M_i is $b/(b + 1)$. For example, the Bayes factor of the model that associates CEPT to HDL versus the model that assumes them unrelated is 140, and implies that the model of dependency between CEPT and HDL is 140 times more likely than the model of independence or, equivalently, that the probability of the dependency between CEPT and HDL is 0.99. Similarly, the Bayes factor 3 between the model that associates LPL with HDL versus the model of independence shows evidence for this association, although the evidence is not very strong.

Furthermore, the program has different algorithms for probabilistic reasoning that can be used to infer the distribution of the phenotype, conditional on a particular genetic profile. Based on this, the program provides validation tools to assess the predictive accuracy of the network, including cross-validation and prediction of the most likely status of one or more network variables in an independent dataset, given values of the other variables. Using one of these algorithms, we assessed the accuracy of the network to predict HDL and LDL cholesterol levels, given information about other clinical variables, exposures, and genetic makeup. The network reached 68% accuracy in predicting HDL and 82% accuracy in predicting LDL using a five-fold cross-validation. The low accuracy in predicting HDL status may be due to the lack of crucial genes that modulate dyslipidemia: this is a typical limitation of candidate gene studies that will be resolved with the availability of more exhaustive information provided by genomewide association studies.

Table 8.1 provides risks for low HDL cholesterol, given different profiles. The table shows the interaction between clinical variables and genetic factors to modulate the risk for low HDL cholesterol level. For example, the same variant of LPL477 is associated with a wide risk range that is modulated by variants on CETP as well as sex, estrogen replacement therapy when applicable, and alcohol consumption.

The network was induced assuming an ordering on the variables based on entropy: less entropic variables can only be parents of more entropic variables. This

Table 8.1 Examples of risk for HDL <40, given genetic profiles and environmental variables.

<i>Profile</i>	<i>Risk for HDL <40</i>	<i>Sex</i>	<i>Estro</i>	<i>BMI</i>	<i>ALC</i>	<i>CETP</i>	<i>LPL447</i>
1	0.52	M	NA	N	No	11	11
2	0.15	F	No	N	No	11	11
3	0.14	F	Yes	N	No	11	11
4	0.46	M	NA	N	No	22	11
5	0.12	F	No	N	No	22	11
6	0.11	F	Yes	N	No	22	11
7	0.30	M	NA	N	0–5	22	11
8	0.05	F	No	N	0–5	22	11
9	0.02	F	Yes	N	0–5	22	11

ordering appears to lead to better models with higher posterior probability. The direction of the arcs, therefore, describes directed association consistent with this convenient order rather than real (physical) associations. However, the directed link from HDL to BMI may seem to be counterintuitive and generate disbelief about the ability of the network to represent the “real mechanism” relating the variables. Although the arcs’ direction describes only a way to factorize a joint probability distribution, in practice providing the undirected graph that is associated with the directed graph may remove the risk of confusion between arc direction and causal processes. The operation of *moralization*—marry the unmarried parents and drop the arc directions—is a simple procedure that maps a directed acyclic graph into an undirected graph and maintains the original Markovian structure of marginal and conditional independence [19]. As an example, Figure 8.9 shows the undirected network derived from the BN in Figure 8.8 by using the “moralization procedure.” Reading the overall set of marginal and conditional independences is based on properties of separations and is described in details in [33] and [34].

8.5 Conclusion

High-throughput technologies are now capable of providing data that can delineate a landscape of the human genome that spans genetic variants, gene expression, and the characterization and measurement of resultant proteins. These technologies are being exploited in long-term cohort studies such as the Framingham Heart Study or the Nurses Health Study, which are collecting longitudinal “phenome-wide” information comprising thousands of variables about thousands of individuals over several decades. But while there is a plethora of data emerging from these studies, quantitative techniques for a holistic analysis of phenome-wide and genome-wide information are lagging behind and seriously limit the pace of discovery.

This chapter has described the use of BN as a tool to describe the genetic basis of complex traits, as a tool for model definition of a complex trait, and their integration. The use of BNs to describe very complex models involving hundreds or thousands of variables requires, however, an investment in computation to adapt current model search algorithms to the specific features of genetic models for com-

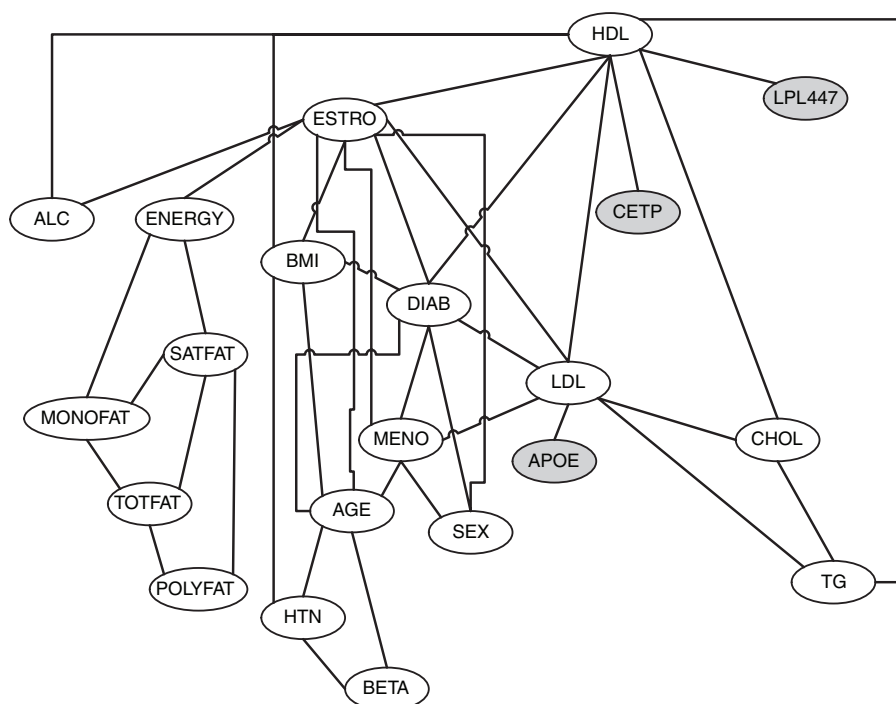


Figure 8.9 The undirected graph obtained from the directed graph in Figure 8.8 by the moralization procedure.

plex traits. For example, algorithms for selection of BNs from SNP data could take advantage of the knowledge about the structure of the genome and patterns of LD, or about gene-gene relations in known pathways, in order to reduce the number of candidate models.

Acknowledgments

The authors acknowledge support from NIH/NHLBI R21 HLO80463-01 and NIH/NIA 5K24AG025727-02.

References

- [1] Botstein, D., and N. Risch, "Discovering genotypes underlying human phenotypes: past successes for Mendelian disease, future approaches for complex disease," *Nat. Genet.*, Vol. 33, Suppl., 2003, pp. 228–237.
- [2] Lander, E. S., and N. J. Schork, "Genetic dissection of complex traits," *Science*, Vol. 265, No. 5181, 1994, pp. 2037–2048.
- [3] Peltonen, L., and V. A. McKusick, "Genomics and medicine. Dissecting human disease in the postgenomic era," *Science*, Vol. 291, No. 5507, 2001, pp. 1224–1229.
- [4] The International Hapmap Consortium, "The international HapMap project," *Nature*, Vol. 426, No. 18, 2003, pp. 798–796.

- [5] Steinberg, M. H., et al., *Disorders of Hemoglobin: Genetics, Pathophysiology, and Clinical Management*, 1st ed., Cambridge, UK: Cambridge Univ. Press, 2001.
- [6] Brookes, A. J., "The essence of SNPs," *Gene*, Vol. 234, No. 2, 1999, pp. 177–186.
- [7] Strauch, K., et al., "How to model a complex trait. 1. General considerations and suggestions," *Hum. Hered.*, Vol. 55, No. 4, 2003, pp. 202–210.
- [8] Strauch, K., et al., "How to model a complex trait. 2. Analysis with two disease loci," *Hum. Hered.*, Vol. 56, No. 4, 2003, pp. 200–211.
- [9] Kroymann, J., and T. Mitchell-Olds, "Epistasis and balanced polymorphism influencing complex trait variation," *Nature*, Vol. 435, No. 7038, 2005, pp. 95–98.
- [10] Daly, M. J., et al., "High-resolution haplotype structure in the human genome," *Nat. Genet.*, Vol. 29, No. 2, 2001, pp. 229–232.
- [11] Zondervan, K. T., and L. R. Cardon, "The complex interplay among factors that influence allelic association," *Nat. Rev. Genet.*, Vol. 5, No. 2, 2004, pp. 89–100.
- [12] Phillips, T. J., and J. K. Belknap, "Complex-trait genetics: emergence of multivariate strategies," *Nat. Rev. Neurosci.*, Vol. 3, No. 6, 2002, pp. 478–485.
- [13] Risch, N. J., "Searching for genetic determinants in the new millennium," *Nature*, Vol. 405, No. 6788, 2000, pp. 847–856.
- [14] Cardon, L. R., and J. I. Bell, "Association study designs for complex diseases," *Nat. Rev. Genet.*, Vol. 2, No. 2, 2001, pp. 91–99.
- [15] Hoh, J., and J. Ott, "Mathematical multi-locus approaches to localizing complex human trait genes," *Nat. Rev. Genet.*, Vol. 4, No. 9, 2003, pp. 701–709.
- [16] Jewell, R., *Statistics for Epidemiology*, Boca Raton, FL: CRC/Chapman & Hall, 2003.
- [17] Gabriel, S. B., et al., "The structure of haplotype blocks in the human genome," *Science*, Vol. 296, No. 5576, 2002, pp. 2225–2229.
- [18] Whittaker, J., *Graphical Models in Applied Multivariate Statistics*, New York: John Wiley & Sons, 1990.
- [19] Cowell, R. G., et al., *Probabilistic Networks and Expert Systems*, New York: Springer Verlag, 1999.
- [20] Ott, J., *Analysis of Human Genetic Linkage*, Baltimore: Johns Hopkins Univ. Press, 1999.
- [21] Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, New York: Springer-Verlag, 2001.
- [22] Nelson, M. R., et al., "A combinatorial partitioning method to identify multilocus genotypic partitions that predict quantitative trait variation," *Genome Res.*, Vol. 11, No. 3, 2001, pp. 458–470.
- [23] Ritchie, M. D., et al., "Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer," *Am. J. Hum. Genet.*, Vol. 69, No. 1, 2001, pp. 138–147.
- [24] Ritchie, M. D., L. W. Hahn, and J. H. Moore, "Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity," *Genet. Epidemiol.*, Vol. 24, No. 2, 2003, pp. 150–157.
- [25] Zhang, H., et al., "Tree-based linkage and association analyses of asthma," *Genet. Epidemiol.*, Vol. 21, Suppl. 1, 2001, pp. S317–S322.
- [26] Chen, C. H., et al., "A genome-wide scan using tree-based association analysis for candidate loci related to fasting plasma glucose levels," *BMC Genet.*, Vol. 4, Suppl. 1, 2003, p. S65.
- [27] Beaumont, M. A., and B. Rannala, "The Bayesian revolution in genetics," *Nat. Rev. Genet.*, Vol. 5, 2004, pp. 251–261.
- [28] Perls, T., "The different paths to age one hundred," *Ann. NY Acad. Sci.*, Vol. 1055, 2005, pp. 13–25.

- [29] Perls, T. T., "The different paths to 100," *Am. J. Clin. Nutr.*, Vol. 83, No. 2, 2006, pp. 484S–487S.
- [30] Perls, T., L. Kunkel, and A. Puca, "The genetics of aging," *Curr. Opin. Genet. Dev.*, Vol. 12, No. 3, 2002, pp. 362–369.
- [31] Perls, T., and D. Terry, "Genetics of exceptional longevity," *Exp. Gerontol.*, Vol. 38, No. 7, 2003, pp. 725–730.
- [32] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Francisco: Morgan Kaufmann, 1988.
- [33] Lauritzen, S. L., *Graphical Models*, Oxford: Clarendon Press, 1996.
- [34] Lauritzen, S. L., and N. A. Sheehan, "Graphical models for genetic analysis," *Statist. Sci.*, Vol. 18, No. 4, 2004, pp. 489–514.
- [35] Sebastiani, P., M. M. Abad, and M. F. Ramoni, "Bayesian networks for genomic analysis," in *EURASIP Book Series on Signal Processing and Communications*, E. R. Dougherty, et al. (eds.), New York: Hindawi Pub., 2005, pp. 281–320.
- [36] Heckerman, D., "Bayesian networks for data mining," *Data Mining and Knowledge Discovery*, Vol. 1, No. 1, 1997, pp. 79–119.
- [37] Kass, R. E., and A. E. Raftery, "Bayes factor," *J. Am. Statist. Assoc.*, Vol. 90, 1995, pp. 773–795.
- [38] Pritchard, J. K., and N. J. Cox, "The allelic architecture of human disease genes: common disease-common variant, or not?" *Hum. Mol. Genet.*, Vol. 11, No. 20, 2002, pp. 2417–2423.
- [39] Hand, D. J., H. Mannila, and P. Smyth, *Principles of Data Mining*, Cambridge, MA: MIT Press, 2001.
- [40] Cooper, G. F., and G. F. Herskovitz, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, Vol. 9, 1992, pp. 309–347.
- [41] Sebastiani, P., et al., "Genetic dissection and prognostic modeling of overt stroke in sickle cell anemia," *Nat. Genet.*, Vol. 37, No. 4, 2005, pp. 435–440.
- [42] Heckerman, D., D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combinations of knowledge and statistical data," *Mach. Learn.*, Vol. 20, 1995, pp. 197–243.
- [43] Dougherty, E. R., R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *International Conference on Machine Learning*, San Francisco: Morgan Kaufmann, 1995.
- [44] Quinlan, J. R., "Improved use of continuous attributes in C4.5," *J. Artificial Intell. Res.*, Vol. 4, 1996, pp. 77–90.
- [45] Fayyad, U. M., and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *IJCAI, AAAI*, 1993.
- [46] Cardon, L. R., and L. J. Palmer, "Population stratification and spurious allelic association," *Lancet*, Vol. 361, No. 9357, 2003, pp. 598–604.
- [47] Friedman, N., D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Mach. Learn.*, Vol. 29, 1997, pp. 131–163.
- [48] Vapnik, V., *Statistical Learning Theory*, New York: J. Wiley, 1998.
- [49] Arking, D. E., et al., "Association between a functional variant of the KLOTHO gene and high-density lipoprotein cholesterol, blood pressure, stroke, and longevity," *Circ. Res.*, Vol. 96, No. 4, 2005, pp. 412–418.
- [50] Ordovas, J. M., and E. J. Schaefer, "Genes, variation of cholesterol and fat intake and serum lipids," *Curr. Opin. Lipidol.*, Vol. 10, No. 1, 1999, pp. 15–22.
- [51] Ordovas, J. M., "Pharmacogenetics of lipid diseases," *Hum. Genomics*, Vol. 1, No. 2, 2004, pp. 111–125.

PART V

Design: Synthetic Biology

Fundamentals of Design for Synthetic Biology

Cody Wood and Gil Alterovitz

9.1 Overview

Recent developments in the study of biological cellular systems and biotechnology have turned the field of bioengineering toward a new direction: synthetic biology.

Synthetic biology is the construction of artificial biological systems, such as programmable cells, using principles from engineering, computer science, and computational biology. This is accomplished either by designing new components that do not already exist in the natural world or by redesigning natural biological systems [1, 2]. Through the construction of biological parts and systems, synthetic biologists learn about natural biology by simplifying and redesigning natural systems. The ability to design cellular systems for specific functions and applications is revolutionizing molecular biology. Synthetic biologists are engineering complex, predictable, and reliable cells into living devices that function as molecular factories by embedding biochemical logic circuits and intercellular communications into cells [3].

There have been numerous recent advancements in synthetic biology. In academia, the Registry of Standard Biological Parts has been developed based on the concept of BioBricks, allowing scientists to browse through standardized and interchangeable biological parts for building synthetic biological systems. In the corporate world, Codon Devices is aiming to commercialize DNA synthesis on demand. Another significant advancement in the field is the first description of the RNA riboregulator and other biological circuit components such as the genetic toggle switch. This progress has led to the development of engineered simple genetic circuits that mimic other common devices like oscillators and feedback loops [3]. Other recent achievements include the development of non-native behaviors like optimized drug synthesis and programmed spatial formation [4].

Although scientists have made significant progress in the development of synthetic biology, the field must still overcome several challenges. The inherent complexity of natural biological systems, the lack of standardization for the construction of reliable synthetic biological systems, and natural evolution are problems that make progress costly and laborious [5]. Scientists are working vigorously

on the following specific areas to advance synthetic biology and overcome the great challenges of the field:

1. The creation of a registry of standard biological parts to make the design and construction of synthetic circuits easier and more effective.
2. The combination of synthetic components to create complex biological circuitry to function in vivo (within the living system).
3. The engineering of multicellular systems to obtain complex, predictable, and reliable cell behaviors by embedding logic circuits and programming intracellular communication.
4. The directed evolution of synthetic genetic circuits to optimize performance for a diverse set of applications.
5. Engineering the interface between biological systems and engineered systems to bridge the gap between natural and synthetic life.

9.2 Circuits

Like electric circuit designers, biological circuit designers begin with a simple circuit. Biological circuits are assemblies of genes and regulatory DNA that act as the biochemical equivalent of electronic components; proteins are the wires and genes are the gates (see Figure 9.1). These genetic components must be well characterized, function similarly in different systems, and act independently of other cellular processes [6].

Early experimental circuits focused on the novel transcriptional cascade because the mechanisms of transcription are relatively well understood. The concept is derived from transcription, which is the process of copying the sequence of DNA to RNA by the enzyme RNA polymerase to transfer genetic information to a form that can be directly used to synthesize proteins in cells [7, 8]. To calculate the concentrations of mRNA $[M]$ and proteins $[P]$ during gene expression, the following kinetic rate equations can be used:

$$\begin{aligned}\frac{d[M]}{dt} &= \frac{k_{on}}{k_{on} + k_{off}} \frac{s_A}{V} + \frac{k_{off}}{k_{on} + k_{off}} \frac{s_R}{V} - \delta_M[M] \\ \frac{d[P]}{dt} &= s_P[M] - \delta_P[P]\end{aligned}$$

where V is cell volume, $\delta_M[M]$ and $\delta_P[P]$ are the degradation rates for mRNA and proteins, and $s_P[M]$ is the rate of protein synthesis. The rate constants, k_{on} and k_{off} govern transitions between the active and inactive states of the promoter. The terms s_A and s_R are the rates of activated and repressed RNA synthesis [9].

The biological circuit is composed of genes arranged in series, in which each gene regulates the expression of one downstream target. In transcriptional cascades, when one gene is transcribed, it ultimately results in the synthesis of a protein that may induce the transcription of other genes that do the same. Essentially, each unit is connected so that the output of one is the input of the next.

Transcriptional cascades direct temporal programs of successive gene expression. An initial signal can be amplified substantially with each downstream element

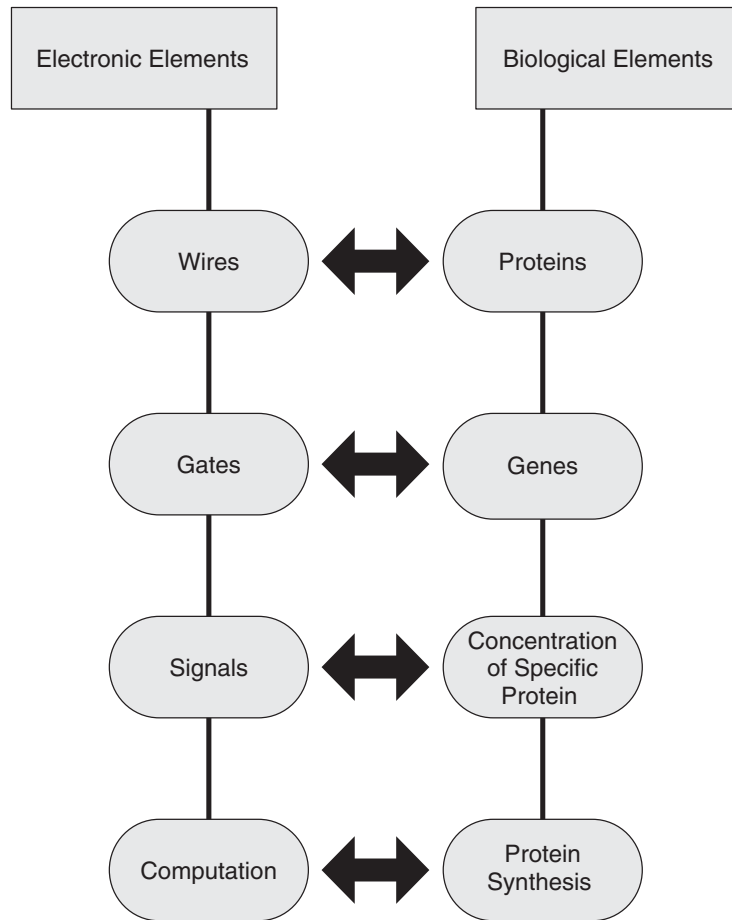


Figure 9.1 Biological versus electronic circuit comparison.

in transcriptional cascades. Because there are many steps in transcriptional cascades, there are many potential opportunities for regulation and integration with other signals from different pathways.

The next development in synthetic biology was the biological inverter, designed like the inverter in electric circuits (see Figure 9.2). The basic parts include a repressor, ribosome binding site, terminator, and an operator. If the input repressor is absent, RNA polymerase transcribes the gene to synthesize the output protein. If the input repressor is present, no output protein is synthesized [8].

More specifically, the inverter begins as a polymerase molecule that transcribes a DNA strand starting with the inverter's input and ending with a sequence that stops the polymerase from transcribing. This produces a control protein. The control protein binds to a "landing pad" for polymerase near the inverter's output, blocking other polymerase molecules from latching on and transcribing DNA. If the "landing pad is clear," a free-floating polymerase molecule latches on and begins transcribing at the inverter's output, continuing down the DNA strand.

When no input protein is present, the gene is turned on and the encoded protein is produced. When the input protein is abundant the gene turns off and there

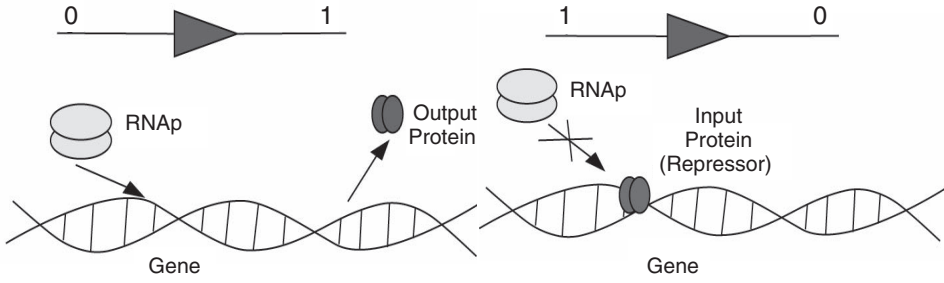
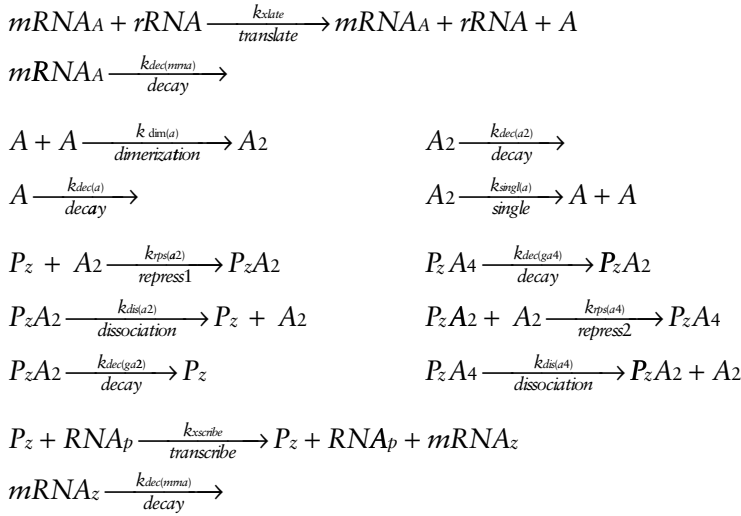


Figure 9.2 Example of a biological inverter. (After: [16].)

is no protein synthesis. The following is a set of biochemical reactions that can be used to mathematically model the kinetics of an inverter, taking into account the rates of transcription and translation, dimerization, dissociation, and decay [10].



where $mRNA_A$ is the input and $mRNA_z$ is the output; A is a protein repressor, and A_2 is the dimeric form of A . P_z denotes the concentration of the active form of the promoter for Z whose operator is unbound by a repressor. Thus, $P_z A_2$ and $P_z A_4$ are the repressed forms of the promoter. RNA_p is RNA polymerase, which transcribes the unbound promoter, P_z , into $mRNA_z$, the gene transcript that codes for other signals.

The inverter allowed scientists to progress to more complex circuitry and introduce more advanced components [6]. These components can be created in vitro using predictive models and inserted into a cell in which the circuit behaviors can be identified. Component examples include riboregulators, feedback loops, toggle switches, logic gates, and oscillators.

9.2.1 Riboregulators

RNA riboregulators are nucleic acid elements that enable scientists to control protein production and give insight into the mechanical actions of RNA-based processes [11]. The ability of RNA to adopt complex structures gives it unique and specialized behaviors. This structure-function relationship affects the interaction between RNA

and numerous other molecules, including proteins, metabolites, and nucleic acids. Due to these specialized interactions, RNA can be engineered to respond to specific nucleic acids and proteins to precisely control gene regulation [12].

The riboregulator consists of a sequence of DNA that integrates into the host bacterium's genome via a genetically engineered virus. In the bacterium, the DNA creates a loop of mRNA that binds to a site on the ribosome. Since the ribosome is responsible for protein production in the cell, the bound mRNA blocks the production of a specific protein. Riboregulators can also unblock the ribosome on command to continue protein production [7].

Riboregulators offer the capability of programming cell behavior and genetic networks with respect to cellular state and response to environmental stimuli. They can also be designed to regulate the expression of any target transcript in response to any ligand (see Figure 9.3) [13].

One example of a riboregulator is the allosteric aptamer construct, or ligand-controlled riboregulator. Apatamers, also referred to as “antiswitches,” are a nucleic acid species that binds specific ligands and imparts allosteric control properties to other functioning RNA molecules. This behavior allows for the construction of in vitro signaling aptamers. In the allosteric aptamer construct, the aptamer interacts with the protein transcriptional activators to induce transcription after binding with the dye tetramethylrosamin [14].

Engineered riboregulators have fast response times to biological stimuli and may prove useful as cellular sensors capable of detecting fluctuations in biological signals. Additionally, due to the highly specific base-pair interactions of RNA, RNA probes could be engineered to reveal functional properties of large networks as well as specific proteins [12].

9.2.2 Feedback Loops

Feedback loops are also used in the construction of biological circuits. They are autoregulatory systems that allow a protein to modify its own rate of production by directly or indirectly affecting the rates of transcription. Whether the feedback is positive or negative depends on the network dynamics of the cell. Negative feedback tends to slow a process and help maintain stability, whereas positive feedback

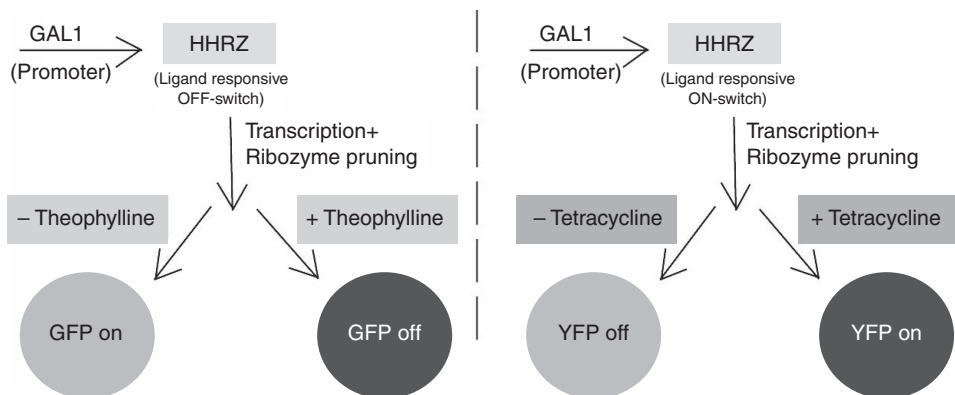


Figure 9.3 Example of an engineered ligand-controlled riboregulator. (After: [13].)

tends to accelerate it and amplifies divergence possibilities. One key trait of positive feedback is that it generates bistability, or two steady states. Bistability is the minimum requirement for a network to have memory [15].

9.2.3 Toggle Switches

Another circuit component is the toggle switch. Like a transistor, the toggle switch acts as an on/off function [11]. The toggle switch also creates bistability but, unlike feedback loops, uses the principle of mutual repression. One example of this in nature is the natural switch from bacteriophage λ used to control the lysis/lysogeny decision.

The criteria for the design of toggle switches include strong and balanced constitutive promoters, the formation of protein multimers, and similar degradation rates for the two main components. One synthetic genetic toggle switch uses the promoters P_{trc}-2 and P_{LS1con}, whose protein products repress the transcription of the other promoter [15].

9.2.4 Logic Gates

Logic gates are another important component for the construction of synthetic biological circuits. Biological logic gates are analogs of logic gates in electronic circuits. One example of this device can be seen through the following process: input protein/inducer pairs to a regulated promoter. The output is on if the gene downstream of the promoter is being transcribed, and off otherwise [11]. The use of and/or gates could lead to even more complex circuitry. With optimized gates, the input/output will have the desired characteristics for constructing a robust genetic logic circuit [16].

9.2.5 Oscillators

Oscillators or clocks like the ones found in nature are also useful tools for the production of synthetic life. Autonomous oscillators are found in gene expression and metabolic, cardiac, and neuronal systems. Circadian rhythms manifest themselves based on the variation in concentration of a particular protein.

From nature, *de novo* (newly synthesized) oscillators are produced and used to explore potential applications [17]. The study of these systems shows the possibility of metabolic flux as a controlling factor in system-wide oscillation. Several synthetic genetic clocks have been constructed in bacteria, including the Repressilator in *Escherichia coli*, which periodically induces synthesis of a fluorescent protein to indicate cell state [18]. The circuits are much simpler than the clocks found in *Drosophila* but are the first step in gaining further insight into the dynamics of the actual organism [4].

9.3 Multicellular Systems

Synthetic circuits have also been created for intercellular communication. Coordinated aggregate cell behavior is often the result of chemical diffusion to carry in-

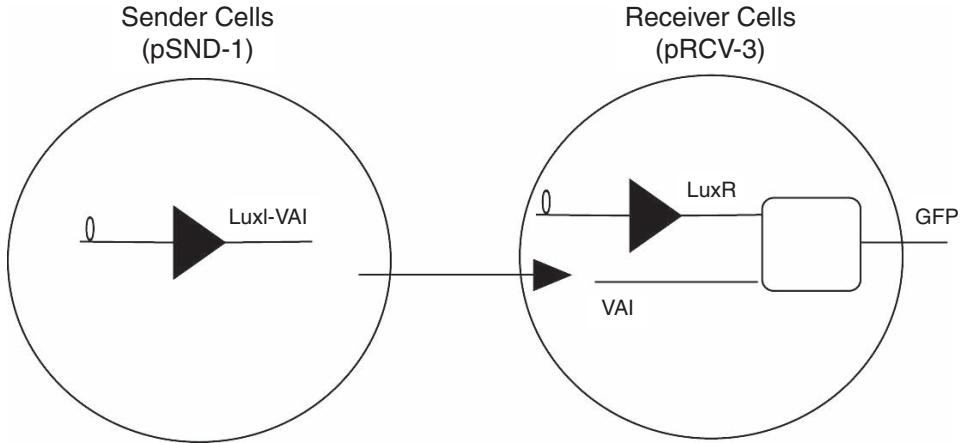


Figure 9.4 Example of a genetic logic circuit. (After: [10].)

formation [10]. This synthetic multicellular system involves genetically engineered “sender” and “receiver” cells (see Figure 9.4). The receiver cells are programmed to form by using line patterns based on the chemical gradient of acylated homoserine lactones (AHL) signals synthesized by sender cells. The construction of multicellular systems improves scientists’ quantitative understanding of natural processes [19]. The systems have the potential to foster applications in tissue engineering, bio-material fabrication, and biosensing.

However, the design of the multicellular system is not as simple as sender and receiver cells. First, the receiver cells are programmed to form ringlike patterns of differentiation based on the chemical gradients of AHL. Only receiver cells at intermediate distances from sender cells express the output protein [20]. The communication from the senders is initiated by the expression of the LuxL enzyme, which catalyzes the synthesis of AHL. The AHL diffuses to nearby receiver cells and is bound by LuxR. LuxR is an AHL-dependent transcription regulator and activates the expression of λ repressor Cl and Lac repressor [16]. The following equations were used to model the activation and repression of protein synthesis by Basu et al. [19].

$$\frac{dG}{dt} = \frac{\alpha_G}{1 + (L / \beta_L)^{\eta_1}} - \gamma_G G$$

$$\frac{dL}{dt} = \frac{\alpha_{L1}}{1 + (C / \beta_C)^{\eta_2}} - \frac{\alpha_{L2} R^{\eta_3}}{(\theta_R)^{\eta_3} + R^{\eta_3}} - \gamma_L L$$

$$\frac{dC}{dt} = \frac{\alpha_C R^{\eta_3}}{(\theta_R)^{\eta_3} + R^{\eta_3}} - \gamma_C C$$

$$\frac{dR}{dt} = \rho_R [\text{LuxR}]^2 A^2 - \gamma_R R$$

$$\frac{dA_{x,y,z}}{dt} = \varepsilon (A_{x-1,y,z} + A_{x+1,y,z} + A_{x,y-1,z} + A_{x,y+1,z} + A_{x,y,z-1} + A_{x,y,z+1} - 6A_{x,y,z})$$

where G is [GFP], L is [LacI], CI is [C], R is the [LuxR/AHL complex], and A is [AHL]. The concentration of LuxR is fixed, θ_R is the LuxR/AHL activation coefficient, ρ_R represents LuxR/AHL dimerization, α and β are rates of synthesis and repression, and γ is the rate of protein decay.

The formation of complexes between the proteins and the ligands in the process is fundamental to the biological process at the molecular level. The ability to manipulate the complexes is critical for the development and understanding of the systems. The systems may also unveil the processes for natural pattern formation, which is the main feature of coordinated behavior involving cell-to-cell communications [16].

9.4 Challenges

9.4.1 Standardization

One problem with the development of circuit components is the lack of standardization. One solution being developed is the concept of BioBricks, a system of defined components that enables synthetic biologists to exchange parts, assemble subcomponents, outsource assembly, and rely on previously made components. OpenWetWare, an online registry of standard biological parts and other resources, was created by MIT in an effort to promote the sharing of information among researchers and groups who are working in biology and biological engineering, based on the initial BioBricks concept. Like Open Software, OpenWetWare hopes to standardize the biological parts repository and spur innovation and improvement of existing biological circuit components.

BioJade, a design environment based on Java, includes the BioBricks parts repository and has created the next generation of bioengineering and computer engineering through standardized molecular control [21]. This standard represents a modular way of building and composing parts made of DNA. The parts are characterized by known maximum signal levels, transfer functions, transcription load on the organism, and cross-talk. Although there are biological limitations like apoptosis, necrosis, and slow diffusion and transcriptional processes, the potential applications are vast. Biological energy production and storage are just two applications for which the BioJade environment may help engineers and scientists revolutionize molecular biology.

9.4.2 Stochasticity

Another difficulty in the design of synthetic biological circuits is the stochasticity of natural cells. Two cells exposed to the same environment can vary greatly due to the stochasticity in gene expression. The difference between the cell's gene expressions arises from fluctuations in transcription and translation. This cellular trait gives the cells the flexibility to adapt but creates problems in the development of synthetic cellular systems [9]. Although computer simulation and mathematical modeling allow scientists to rationally predict the behavior of synthetic circuits, there is currently no way to know how the circuit will behave within a natural system. One way scientists are seeking to control the problem is through directed evolution. This

method stems from construction that relies on the cell's ability to survive under specific pressures [22].

9.4.3 Directed Evolution

Directed evolution is becoming a more common technique that applies specific stress to force the system to produce the desired outcome. This allows biological circuits to tolerate certain bad design features by forcing the system to evolve. Biological circuit engineers use computer simulation and mathematical modeling to rationally design circuit behaviors, but they are unable to predict the precise behavior of synthetic networks and therefore need a design strategy to correct the in vivo behavior changes. Construction of the circuit in vivo currently requires optimization of biological parameters—such as protein-DNA interactions—that are as yet too poorly understood [23].

The solution is directed evolution. This elementary process begins with the design of a well-understood engineered circuit from a set of devices with well-characterized device physics. Once the circuit is implanted it may be rendered nonfunctional due to the context-dependent behavior of the biological components of the circuit. The circuit is then tuned in vivo using laboratory methods for molecular optimization. It has been demonstrated that by applying directed evolution to genes comprising a genetic circuit, a nonfunctioning circuit of mismatched components can rapidly evolve into a functional circuit. Sequential rounds of random mutagenesis, recombination, and high-throughput screening have proven effective for the modification of individual proteins as well as metabolic pathways [24].

The technique has also become common for the engineering of enzymes, which are natural catalysts, for a diverse set of applications. One problem is that specific enzyme modifications frequently demand an unattainable understanding of the relationship between sequence and frequency. Directed evolution bypasses the problem by combining mutagenesis with selection or screening to identify the improved variant [9].

9.4.4 Random and Targeted Mutagenesis and Recombination

Random mutagenesis, targeted mutagenesis, and recombination are common techniques for the practice of directed evolution. Random mutagenesis can be used to discover beneficial mutations by using a function similar to the desired function to randomly mutate the full gene of an enzyme. The improved variants are screened following the use of polymerase chain reaction (PCR) to average the mutations. The altering of an enzyme's specificity or regioselectivity requires multiple mutations of a specific or active site.

Targeted mutagenesis can also be used to discover beneficial mutations, but unlike random mutagenesis, targeted mutagenesis requires structural or biochemical information. This additional information enables for the targeting of specific active site residues, which is necessary for mutations that are beneficial only after several generations [9].

Another technique within directed evolution is the recombination of structurally similar proteins, which allows for access to larger degrees of sequence change

than mutagenesis. These three techniques are established and effective methods for the engineering of enzymes and have been applied to the development of intein-based molecular switches that transduce the binding of a small molecule to the activation of an arbitrary protein [25]. However, both recombination and mutagenesis still have imperfections. As structural and chemical knowledge grows with our ability to rationally design synthetic circuits, directed evolution will become a very powerful tool [9].

9.4.5 System Interface

The system interface is also a critical area to control to improve the predictability of synthetic biological systems. Other systems use modularity to make interacting systems interchangeable and insulate one from another. Engineered biosystems are embedded in complex and variable host cells. The chance of success is improved if the system function is decoupled from the state of the host cell. This will lead to the interchange between different chassis. Synthetic biological systems also rely on the host cell for the processes of transcription, translation, degradation, and the requisite energy and materials to power those procedures [26].

9.4.6 Kinetics

Another major challenge to synthetic biology involves altering the kinetics of each individual element of the circuit so they are impedance matched to function correctly in the context of the network. In the human body, evolution has optimized each component. This idea of genetic process engineering is used to modify the DNA of existing genetic elements until the desired behavior for implementing engineered biological components for complex circuit design is achieved [10]. These genetic modifications should produce components that mimic digital computation. In the same way that a recording engineer filters out noise, synthetic biologists need to screen out the noise created by a biological circuit.

In synthetic biology, large fluctuations in rates are caused by the interaction between the number of transcription and translation rates and the number of promoter sites and mRNA resulting in internal noise [11]. Designing an appropriate selection mechanism that relies entirely on the cell population to choose the best match will help solve the problem [7].

9.5 Conclusion

Although there are several areas hindering the rapid advancement of synthetic biology, the field has incredible potential. Applications include energy production and storage, nanoscale devices, molecular medical devices, bioreactors, programmable devices, smart materials, and sensors (see Figure 9.5).

Around the world, up to three million people die per year of malaria. Although the plant-derived drug artemisinin has a success rate close to 100%, it is too expensive for developing countries. Synthetic biologists have now created a bacteria

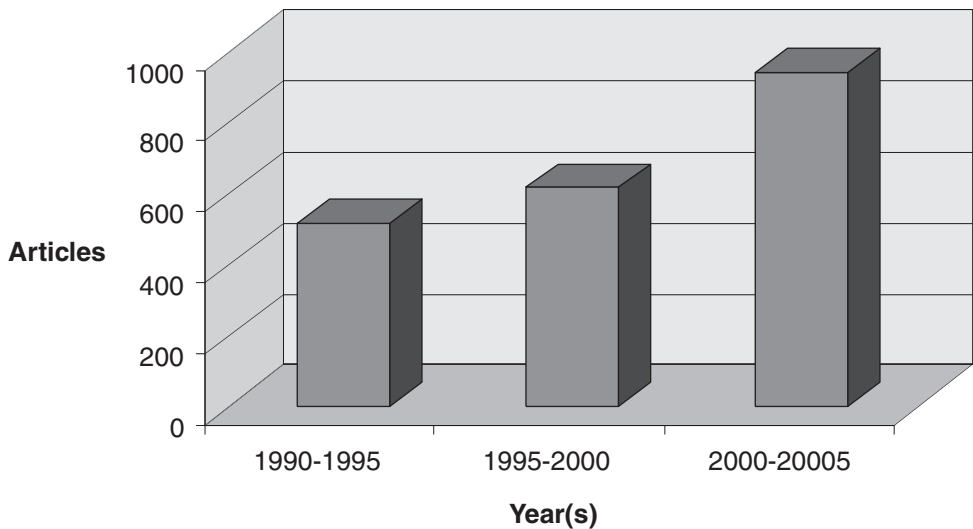


Figure 9.5 PubMed hits for synthetic biology by year.

strain that can produce the precursor to artemisinin by inserting genes from three separate organisms into *Escherichia coli* [27]. If scientists could design the bacteria to produce artemisinin, it would save millions of lives. Synthetic biology could also open the doors for the development of microorganisms that produce hydrogen or convert sunlight energy into chemical forms.

From the simple design of novel biological devices to the ability to extend our immune system, synthetic biology has created new ways to consider the possibilities of life. The unique approach of synthetic biology promises novel solutions to problems natural biology has been unable to resolve and provides scientists with an exciting new approach to understanding life.

References

- [1] Clark, L., et al., “Synthetic biology applies engineering approach to biological components,” *Engineering Our World*, 2004.
- [2] Gibbs, W. W., “Synthetic life,” *Sci. Amer.*, Vol. 290, No. 5, 2004, pp. 74–81.
- [3] Ferber, D., “Microbes made to order,” *Science*, Vol. 303, 2004, pp. 158–161.
- [4] Sam, J., “In the business of synthetic life,” *Sci. Amer.*, Vol. 292, 2005, pp. 40–41, 2p, 1c.
- [5] Endy, D., “Foundations for engineering biology,” *Nature*, Vol. 438, 2005.
- [6] Sprinzak, D., et al., “Reconstruction of genetic circuits,” *Nature*, Vol. 438, 2005.
- [7] McDaniel, R. W. R., “Advances in synthetic biology: on the path from prototypes to applications,” *Curr. Opin. Biotechnol.*, Vol. 16, 2005, pp. 476–483.
- [8] Abelson, H., et al., “Amorphous,” *Commun. ACM*, Vol. 43, 2000, pp. 74–82.
- [9] Kaern, M., et al., “Stochasticity in gene expression: from theories to phenotypes,” *Nat. Rev. Genet.*, Vol. 6, 2005, pp. 451–464.
- [10] Weiss, R., “Cellular computation and communication using engineered genetic regulatory networks,” Ph.D. thesis, MIT, 2001.

- [11] Issacs, F. J., et al. "Engineered riboregulators enable post-transcriptional control of gene expression," *Nat. Biotechnol.*, Vol. 22, 2004, pp. 841–847.
- [12] Issacs, F. J., et al., "RNA synthetic biology," *Nat. Biotechnol.*, Vol. 24, No. 5, 2006.
- [13] Bayer, T. S., et al., "Programmable ligand-controlled riboregulators of eukaryote gene expression," *Nat. Biotechnol.: Advance Online Pub.*, 2005, pp. 1–7.
- [14] Morton, O., "Life, reinvented," in *Wired*, 2005.
- [15] Hasty, J., D. McMillen, and J. J. Collins, "Engineered gene circuits," *Nature*, Vol. 420, 2002, pp. 224–230.
- [16] Weiss, R., and Basu, S., "The device physics of cellular logic gates," In *NSC-1: The First Workshop on NonSilicon Computing*, 2002, pp. 54–61, Princeton Univ.
- [17] Fung, E., et al. "A synthetic gene–metabolic oscillator," *Nature*, Vol. 435, 2005.
- [18] Elowitz, M., and Leibler, S., "A synthetic oscillatory network of transcriptional regulators," *Nature*, Vol. 403, 2000.
- [19] Basu, S., et al., "A synthetic multicellular system for programmed pattern formation," *Nature*, Vol. 434, 2005, pp. 1130–1134.
- [20] Looger, L. L., et al. "Computational design of receptor and sensor proteins with novel functions," *Nature*, Vol. 423, 2003, pp 185–190.
- [21] Goler, J. A., *BioJADE: A Design and Simulation Tool for Synthetic Biological Systems*, 2004, pp. 1–56.
- [22] Blake, J. W., and J. F. Isaacs, "Synthetic biology evolves," *Trends Biotechnol.*, Vol. 22, No. 7, 2004, pp. 1–3.
- [23] Yokobayashi, Y., et al., "Directed evolution of a genetic circuit," *Proc. Natl. Acad. Sci. USA*, Vol. 99, 2002, pp. 16587–16591.
- [24] Bloom, M. M., et al., "Evolving strategies for enzyme engineering," *Curr. Opin. Struct. Biol.*, Vol. 15, 2005, pp. 447–452.
- [25] Buskirk, A. R., et al., "Directed evolution of ligand dependence: small-molecule-activated protein splicing," *Proc. Natl. Acad. Sci. USA*, Vol. 101, 2004, pp. 10505–10510.
- [26] Canton, B., "Engineering the interface between cellular chassis and integrated biological systems," Ph.D. thesis, Biological Engineering Division, MIT, 2005.
- [27] University of California, Santa Barbara, "Synthetic biology," P. B. Division (ed.), 2006, <http://www.lbl.gov/pbd/synthbio/>.

BioJADE: Designing and Building Synthetic Biological Systems from Parts

Jonathan A. Goler and Thomas F. Knight Jr.

10.1 Introduction

This chapter discusses the development and use of BioJADE as a tool for building and analyzing synthetic biological systems. First, it discusses the inspiration of BioJADE, from both synthetic biological examples such as the Repressilator and fundamentals upon which BioJADE is built: standard parts, the BioBricks construction technique, and abstraction. The discussion then focuses on the architecture of BioJADE and follow with an example system design. Finally, there is a discussion of the actual and perceived limitations of composition, modularity, and system design, and the next steps in development.

10.2 Fundamentals of BioJADE and BioBricks Construction

10.2.1 Inspiration

A few sample systems have been instrumental in understanding gene regulation and cell-cell signaling. One such system, the Lux system, has been studied extensively by Greenberg et al. [1–4]. These systems were then used by Knight and Weiss to implement communication and logic gates [5].

Other systems, such as the periplasmic binding proteins investigated and exploited by Hellinga, can be modified computationally to respond to non-native ligands [6, 7], giving a greater variety of inputs for synthetic systems.

Integrating existing systems with new parts expands the ability to build interesting biological systems. The existing templates of control systems in nature give hints about what is possible, while altering and breaking these existing systems yields information on how and why they are constructed in the manner they are. Vilar and Leibler [8–10] have used synthetic biological systems to study noise resistance in oscillatory networks and incorporate more robust measurements into models. These synthetic systems are useful not only for engineering systems, but also for understanding the mechanisms of natural system performance.

One interesting question posed by natural systems is what is the purpose of having a large signaling cascade, with level upon level of phosphorylation in protein kinase signaling, for example. Through the metaphor of an electronic transistor and its associated transfer function and noise characteristics, and the inherently noisy stochastic behavior of cells, it is easier to understand that these longer pathways provide better amplification and noise resistance characteristics in the noisy cellular environment. Thus critical cellular functions are controlled far more precisely than possible with a very simple control output for each function.

10.2.2 The BioBricks Standard

The BioBricks Standard, developed by Thomas F. Knight Jr. [11], provides a modular way of building and composing parts made out of DNA. BioBrick parts are, at a chemical level, simply specific series of DNA bases. As an engineering construct, a BioBrick part encapsulates a biological function, from simple parts such as terminators to complex parts such as binary counters. The definition of a BioBrick can contain information on its performance characteristics, operating environment, and stability in a living system.

The BioBrick standard consists of specific prefixes and suffixes that enable the composition of parts in a standard way. There are currently a few ways of composing parts: the idempotent standard assembly method and the triple antibiotic assembly or 3A method [12]. The 3A method has a key advantage in that it does not require gel extraction, thus increasing the efficiency of ligation and the per-stage success rate. In practice, the original BioBricks construction algorithm has a success rate of 30–80%, depending on the level of experience of the cloners, with a minimal per-stage time of 2 days. With a binary tree construction strategy and no failures, it would take $2 \cdot \log_2 n$ days to construct a system of n components. Of course, in practice, with failures, the process can vary greatly, from $O(2^n)$ days to $O(\log_{3/2} n)$, depending on the failure rate. As of publication, there has been no large-scale deployment of the 3A assembly method, so its success rate cannot be accurately assessed.

10.2.3 BioBrick Definition

The BioBricks parts are comprised of their contents and standard BioBrick ends [13, 14] (see Figure 10.1). The contents are arbitrary, with the caveat that they may not contain any of the BioBrick restriction sites (EcoRI XbaI, SpeI, NotI, and PstI). These sites can be mutated out through manual edits of the sequence, either through site-directed mutagenesis, PCR if the undesired sites are near the ends, or direct synthesis of an optimized sequence. In most cases, changes can be made that do not affect the system due to the redundancy in codon specificity.

The prefix for a part is a cctt + XbaI + g site, and the suffix is a t + SpeI + a + NotI + PstI + cctt. The restriction sites enable the idempotent construction, while



Figure 10.1 BioBrick ends assembly schematic.

the extra bases help to separate restriction sites and allow the enzymes some overhang at the ends.

10.2.4 The Abstraction Barrier

The synthetic biology model of biology envisions a distinct abstraction hierarchy of parts, devices, and systems. To build a system such as a counter, one needs devices such as half-adders, composed of smaller devices (inverters), which themselves are composed of parts such as DNA binding proteins, ribosome binding sites, and promoters. The abstraction barrier is necessary to reduce the workload on people designing and building synthetic biological systems. In order to build a system without an abstraction barrier, one would have to design the system from the ground up, designing or retrieving from nature every single part and device used to compose the system. Instead, by separating the work and defining an interface between the different levels of design, there can be systems designers, who get devices from device designers, who get parts from parts designers. In each of these instances, one can imagine that instead of having custom designed parts for each system, there is a company, Sangamo, that specializes in DNA binding proteins, such as Zinc Finger domains. The resulting quality and quantity of parts and devices will result in a much larger possible complexity for systems.

In designing the abstraction barrier, it is important to realize that the choice of abstraction boundaries is an important nonobvious one. The debate over possible choices for an abstraction with regard to signal carriers is an interesting example. The first and more obvious choice for biologists for the signal carrier for, say, transcriptional devices, is the level of signaling protein. The development and measurements of the first synthetic systems focused on the levels of proteins, both signals and reporters. Inverters, for example, the cI-LacI inverter, were mapped out with a transfer function showing cI in, LacI out, with an inverted relationship between the two. This relationship could be measured by using fluorescent protein reporters as proxies for the levels of each of those proteins.

It is then simple to take three such inverters, $cI \rightarrow LacI$, $LacI \rightarrow tetR$, and string them together on a plasmid, using standard BioBricks construction techniques. Such a system will work. However, assume that there is now an incompatibility in one of the proteins, that cI is poisonous to the new host cell the system is being inserted into. In order to change cI to another protein, X, both the $cI \rightarrow LacI$ inverter and the $tetR \rightarrow cI$ inverter need to be changed, assuming $X \rightarrow LacI$ and $tetR \rightarrow X$ inverters are even available. Instead, if a carrier signal, such as the number of DNA polymerases passing across a point on DNA in a second (polymerase per second, or PoPS), were used, that type of signal would be independent of proteins. The black box surrounding the device can be moved, such that the input and output signals are PoPS. The PoPS drive protein production of an internally consistent protein (any of the myriad ones available).

In effect, PoPS provides a common signal carrier, which can be quantified and measured in a standard assay. In addition, as shown in Figures 10.2 and 10.3, by having PoPS as the signal carrier, the interface between parts is simplified. Various promoters can then be tested by having them produce standard transcript. The resulting quantity of mRNA can be determined via quantitative reverse transcription PCR and, from that, the PoPS value can be deduced.

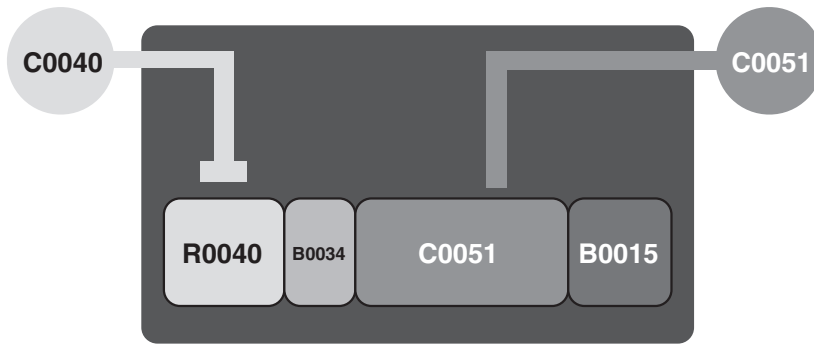


Figure 10.2 A black box diagram showing a transcriptional inverter, which inverts the incoming signal, C0040, and outputs C0051. The inputs and outputs, the proteins C0040, and C0051, are shown as connect points outside the black box. (Courtesy of Drew Endy.)

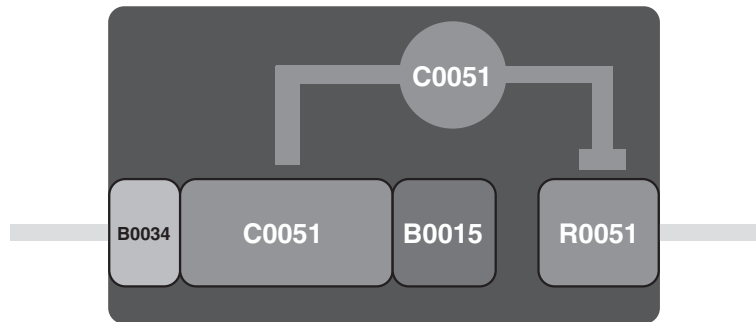


Figure 10.3 A black box diagram showing a transcriptional inverter, but rearranged such that the signal carrying protein, C0051, is wholly contained within the inverter. The input and output of the inverter are both PoPS along the DNA strand. This abstraction is more general in its connectivity: the input and output can be any other PoPS-generating or PoPS-consuming device. (Courtesy of Drew Endy.)

10.3 Representing Parts

DNA, represented as such, while certainly information rich, is not an ideal way to encapsulate the idea of a part, at least for the purpose of engineering a system. One can represent a part in a number of ways. As DNA, a part is the physical *implementation*, exactly as it would appear in a cell. As a symbol, it expresses the *function* of a part, which describes what the part does. Or, as a *diagram*, it is a representation of how the part works.

Even the simplest level of devices can be represented at all three of these aspects. For instance, a promoter, which is a relatively simple part, can be described as its sequence of bases (GCTAGCCGCGTATAAGCGAATTCAGGAGGCCAGGT); as its function, a forward or backward facing arrow; or as a diagram representing the various positions and functions on the promoter (the $-10/-35$ sites, the operator sites for the various proteins that bind to it, and affinity, cooperativity, and interactions associated with it).

While a promoter is a complicated system, a ribosome binding site (RBS) is not generally considered to be as complicated. The RBS is a short sequence, 6–8 bases that are complementary to the Shine-Delgarno sequence on the 16s ribosome. The RBS provides a “landing pad” for the ribosome. The exact sequence of the RBS intimately

affects the output (RiPS, or ribosomes per second) of the RBS. In this case, the melting temperature of the RBS DNA (the physical implementation) is vitally important. The inputs to an RBS are the number of copies (derived from the PoPS of the upstream promoter and the stability of the transcript) and the ribosomes themselves. The RBS performs a process (the binding) on the inputs and produces the output RiPS.

10.3.1 Parts Data Model

Encapsulating the differing representations of parts poses a question: what defines a part or device? Is a part the exact DNA sequence? If so, are parts that are made up of the same function and same internal DNA different if they contain a different end sequence (for BioBricks or BioBricks ++, or direct synthesis without BioBrick ends)?

The answer is not necessarily clear. On one hand, if every difference in a part's ends makes it a different part, there will be a lot of duplicate data to manage, regarding the performance and testing of these myriad different but nearly identical parts. On the other hand, for parts such as RBSs, which are appended to coding regions or prepended by RNA modulating devices, the performance of those combined devices is intimately related to the exact sequence of the final construct. (See Figure 10.4 for a parts database model.)

The dilemma of how to correctly represent a part results in the separation of parts/devices into a more generic prototype: their basic function, how they work, and the functional composition of a part, much like the diagram model of the part.

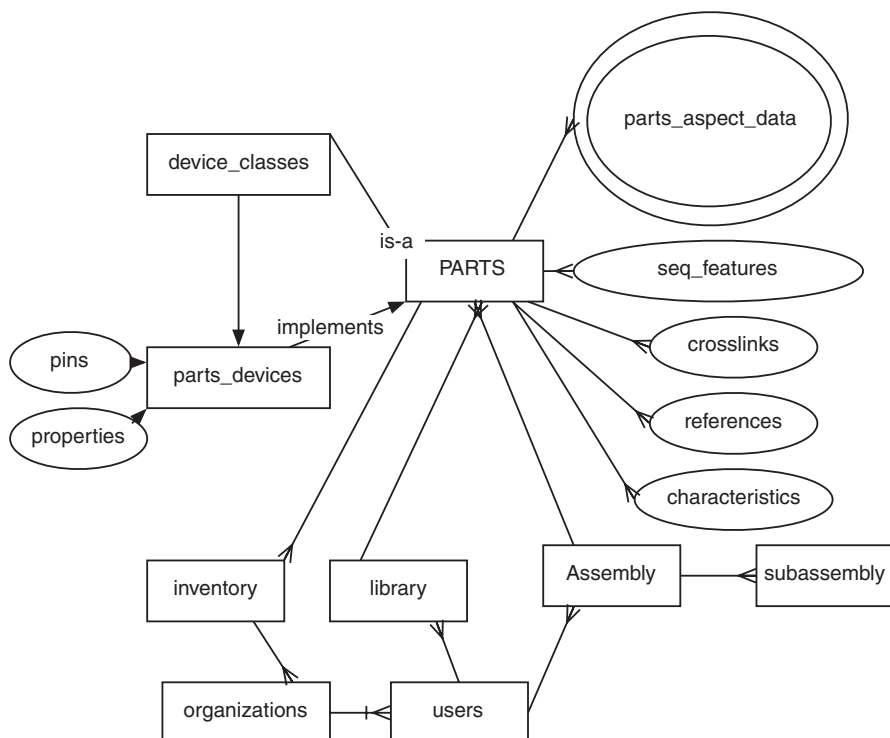


Figure 10.4 Parts database model. The parts table contains basic information, while other tables cover the more variable and external information.

The actual implementation—the hard DNA—must have its own performance characteristics.

10.4 BioJADE Architecture

BioJADE is built on top of the BioBricks part representation model, which includes the design and specification of part features, behaviors, and graphical representations. The system requires one or more repositories for storage and publication of part data.

The system keeps a cache of parts from the data stores and caches changes until they are committed to the database itself. This cache enables fast response time for the user interface. The basic architecture is derived from Jade [15]. However, we use significantly different data structures to handle the differences between silicon and biological substrates.

10.4.1 Aspects

Each part has several aspects, which encapsulate a way of viewing and interacting with a part. The model of aspects was derived from Chris Terman's in Jade. When BioJADE is started, it reads a list of available aspects from the configuration file. These aspects are displayed for the user, and when a part is selected, a user may switch between aspects. To facilitate easy navigation, the aspect abstract class supports saving data in memory for each part so that changes can impact the current workspace without being committed to the data store.

The aspect abstract class essentially handles the loading and saving of aspect data to its data store. Because each aspect has its data represented differently, the individual editors handle actually storing the state to the aspect, which in turn renders it into the data store's XML format.

In addition, the aspects keep track of the current state of all opened BioBricks, so that if you make a change in one part, and then switch to another part, the changes are not lost. This temporary storage is merely a hash table keyed on the BioBrick. This representation is very useful for making changes to parts while not committing them to the database.

The aspects in the Basic BioJADE package consist of the schematic, the icon, the DNA, the functional network, and the simulation.

The *schematic* aspect supports the drag and drop circuit design of more complicated BioBricks. The schematic aspect permits the user to lay out generic devices and then, with a mouse click, “compile” the design into its actual component parts, along with the associated assembly instructions.

The *icon* aspect permits the user to upload a PNG/GIF/JPEG file to display the part and provides a tool for marking the terminals of the device so that it can be connected to others in the schematic mode.

The *DNA* aspect displays the annotated DNA sequence.

The *functional network* aspect shows a more detailed biological-type view of the system, which permits the user to see more rate constants and tune the design.

The *simulation* aspect interfaces to all of the available simulation tools, as well as showing the status of in-progress simulation.

10.4.2 Schematic

The schematic design mode enables users to lay out circuits made out of genetic components just as they would a circuit made of silicon. The design mode presents a toolbar that contains available libraries of parts to stamp onto the design. In addition, a toolbar containing basic functionality such as cut, copy, and paste, and buttons for drawing wires and stamping parts, is docked to the top of the workspace. The rest of the workspace is dedicated to the design itself. Users can scroll and zoom the workspace to focus on different aspects of the design.

Designers build circuits by selecting a part or prototype from the part palette on the left, and selecting the stamp button on the toolbar. The designer then stamps down the part onto the design. The user can then click on the Select button to revert to being able to merely select objects already on the design and move them around.

The designer then selects the Wire tool to draw wires between the components. The first method for drawing wires is to select the starting terminal and drag the wire to the ending terminal. The wires conveniently snap onto the terminals of devices and the terminals of other wires that fall within 10 pixels of their own terminals. In addition, wire drawing is very similar to the way it is done in classic circuit drawing programs. When a wire is initiated (by Alt-click), the user is presented with a draggable wire L and can set the waypoints as she draws. The user is able to flip the wire from going horizontal-vertical to vertical-horizontal and vice versa by typing the “f” key. Additionally, the waypoints are set using the “s” key. The wire is completed by clicking again with the mouse. Clicking with the Alt key held down will set another waypoint. In this manner, the user can use the right hand to draw and the left hand to further modify the line. Finally, the user can press the “s” key to toggle a straight line segment. In addition, the wire drawing mode allows users to Ctl-click on a number of terminals, and hit the “w” key to connect all of the selected terminals with wires.

Once components and wires are laid out, the user can make further adjustments to the design by dragging components to different locations. If a component moves, all connected wires move to accommodate the new position. The blank part templates that are laid out can be assigned specific BioBricks. The designer simply Alt-clicks on the part and selects the BioBrick from the list of matching BioBricks. Once assigned by the designer, those parts are exempted from the compilation optimization process. The user can then save the design to the database by clicking the Save button on the toolbar. The user can also set basic properties of the design, such as a description, design notes, or the target host organism.

After laying out the basic design, the designer clicks the Compile button, selects the appropriate optimizations, and compiles the design into its BioBrick components. Should the compiler run out of assignable parts, it will report the problem to the user. The user could then develop or find libraries of additional parts to complete the design.

The completed compiled design is then reported to the user and can be saved to the database. The user can then go to the functional network mode to fine-tune the DNA sequences and specify additional functional interactions that are not present or tunable in the design mode.

10.4.3 Functional Network Aspect

The functional network aspect and the DNA aspect share several similarities in representation, but where the DNA aspect merely permits the user to view various aspects of the DNA sequence itself, the functional network mode enables the user to edit parts of the sequence, rearrange parts, and change subparts such as ribosome binding sites and terminators (see Figure 10.5).

The DNA is laid out vertically left to right as one goes through the circuit. The designer can click on a segment of DNA and edit its various components to optimize binding strength, or change terminator efficiency. Designers can also physically change the locations of certain genes on the plasmid for whatever reasons might be desirable, such as balancing low-GC regions or exploiting terminator inefficiency to cheat the modularity abstraction, or in order to optimize the local environment of certain genes to improve coexpression.

10.4.4 The DNA Aspect

The DNA aspect contains both the DNA sequence and a parts list, which it uses to display the entire system in a linear fashion (see discussion at the end of this section and Figure 10.9). The parts list contains a linear list of the parts that make up the system. The resulting DNA can be assembled via direct synthesis (copying and pasting the sequence into a synthesis company's website) or through BioBrick assembly. Once compiled, the parts list is inserted into the database; when transferred back to the BioBricks registry, it can be assembled there, where it will be optimized by the inclusion of preconstructed subparts, reducing assembly time.

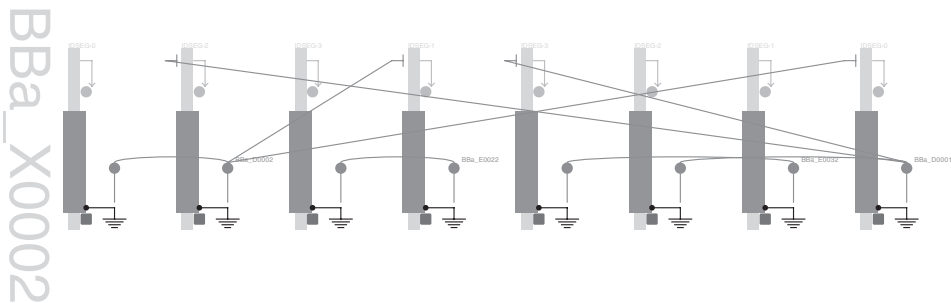


Figure 10.5 A functional network aspect view. Each vertical light gray bar is a contiguous region of DNA, with promoters at the top, RBSs represented as light gray circles, coding regions as large rectangles, and the terminators as dark gray squares. The proteins generated are represented by the dark gray circles coming off of the end of coding regions, with ground signs representing degradation, and lines connecting the proteins with each other for interactions and promoter binding.

10.4.5 Icon Aspect

The icon aspect contains the data on how to incorporate the designed system into larger designs. It allows the selection of an image to represent the part, and the placement of terminals on that part, such that the part can be dropped onto another design and connected to other parts.

10.4.6 Part Repositories

BioJADE is built to use two forms of part repositories; the initial and preferred method is a relational database management system (RDBMS) such as MySQL. The second method is an XML repository stored on the file system. In future versions, BioJADE may support a “web of registries” model where it can access a variety of XML registries via HTTP. Installing the registry as a relational database requires acquisition and installation of an RDBMS, which is often a nontrivial matter.

Originally, BioJADE and the BioBricks registry were built with the same data model. Due to the need for additional flexibility and access control, the BioJADE database and the BioBricks registry were separated data-wise, although the data structures themselves are still nearly identical.

Currently, an automated method of updating the BioJADE registries from the most recent updates in the BioBricks registry allows BioJADE users to have access to the latest and most up-to-date parts.

The BioJADE schema is a star schema, illustrated in Figure 10.4, in which the central table of parts contains the basic part information, and links to other tables that contain more detailed data. Those tables contain sequence information, simulation data, contents of the aspects from BioJADE, diagrams, and papers.

10.5 Using BioJADE, an Example: The Repressilator

The Elowitz Repressilator, one of the first working synthetic biological constructs, provides a good opportunity to demonstrate the functionality and design of a system using part abstraction and the tools in BioJADE.

The Repressilator system is a simple 3-inverter ring oscillator (see Figure 10.6). In electrical engineering, such a construct could be used as a clock. The three inverters are laid out such that the output of the first inverter leads to the input of the second inverter, the output of the second leads to the input of the third, and the output of the third leads to the input of the first. Upon initialization with a high signal,

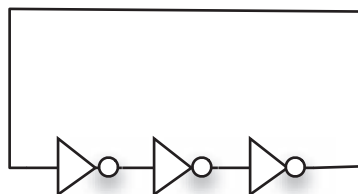


Figure 10.6 Repressilator, as implemented in electrical engineering metaphors. It is composed of three inverters linked in series, in a ring topology.

the first inverter will flip to low, the second to high, and the third to low, which propagates back to the first inverter, flipping the previous high signal to low.

Create a new part, BBa_S0001. BioJADE begins in the blank schematic layout aspect. To build the Repressilator, simply drag three instances of the inverter prototype from the part palette, in a row as in Figure 10.7. Drag wires to connect each of the inverters in the ring topology; the wire terminals change to green when hooked to another component. Save the design and click the compile button. BioJADE goes to its registry and compiles a list of components that could satisfy each of the abstract prototypes in the design. Through the process of constraint propagation, BioJADE eliminates incompatible parts and assigns to each of the prototypes an actual BioBrick device or part based on a combination of minimal cross-talk between components and matching signal levels as close as possible between components. These cross-talk and signal level parameters are kept in the repository based on experimental results.

After compiling, saving the results will keep the resulting concrete parts stored in the design within the repository. Based on the concrete parts, BioJADE will generate the next, more detailed level of the design: the functional network (see Figure 10.8). This more detailed display permits tweaking of the layout of the DNA, along

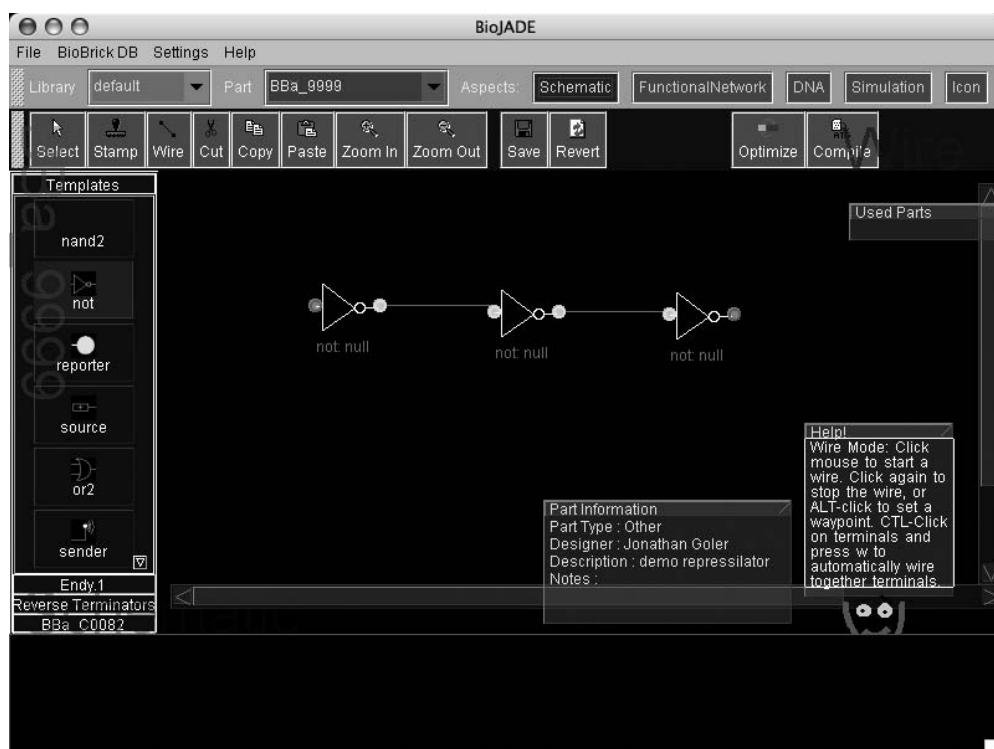


Figure 10.7 Building a Repressilator. Note the designation “not” (inverter) below the parts, and the designation “null.” These parts will become assigned through the compilation process, or can be set manually.

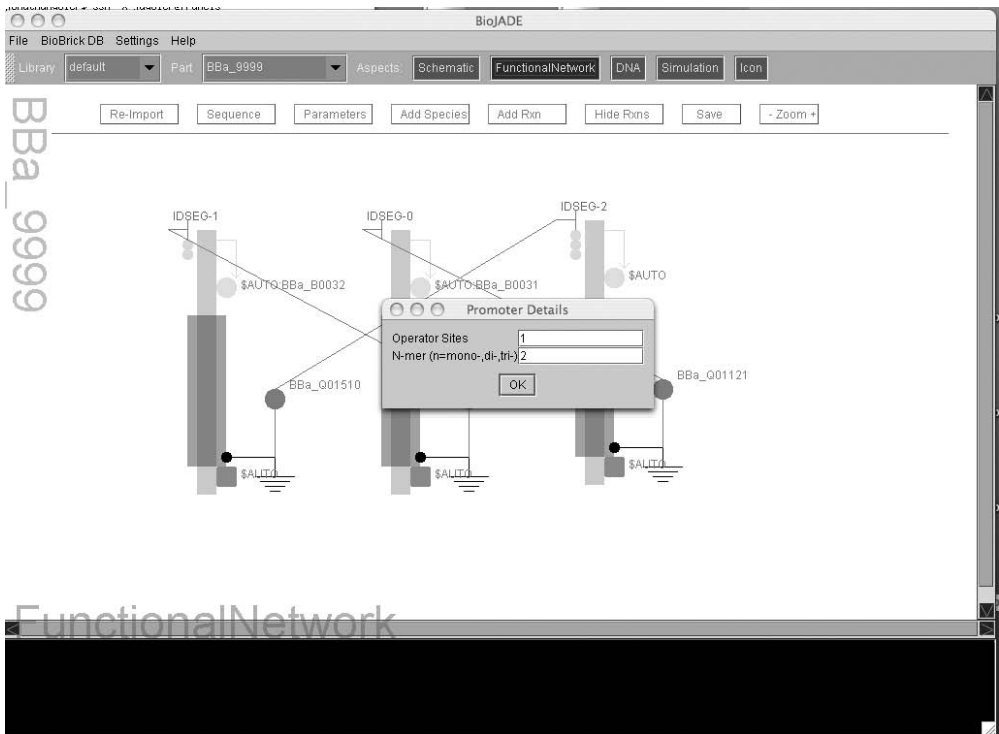


Figure 10.8 The Repressilator in functional network view. The open promoter details box allows the designation of the number of operator sites as well as the cooperativity of the affector molecule, if it was not already specified.

with adjustments to the default ribosome binding sites, cooperativity settings for the simulator, and other factors such as cell growth rate assumptions and copy number for the plasmids.

The design, once specified, can then be passed to the simulators for testing. A matrix of parameters can be entered with which to test the system. These settings can vary both controllable/engineerable parameters—such as strengths of ribosome binding sites, terminator efficiencies, and linear organization of the DNA—as well as less controllable aspects such as copy number and cell growth rate. By varying these parameters, it is possible to understand the sensitivity of the system (while still *in silico*) to variations in those parameters. For instance, by simulating the Repressilator at a variety of copy numbers, it is easy to see just how important having a reasonably high copy number is to prevent the stochastic nature of the cell from disrupting the function of the Repressilator. This behavior is consistent, not entirely numerically, but certainly qualitatively, with experimental data. In the final “output” of BioJADE’s compilation of the Repressilator design (see Figure 10.9), the DNA aspect displays the full DNA sequence of the parts, as assembled, and an annotated strip of DNA showing the parts along a stretch of DNA.

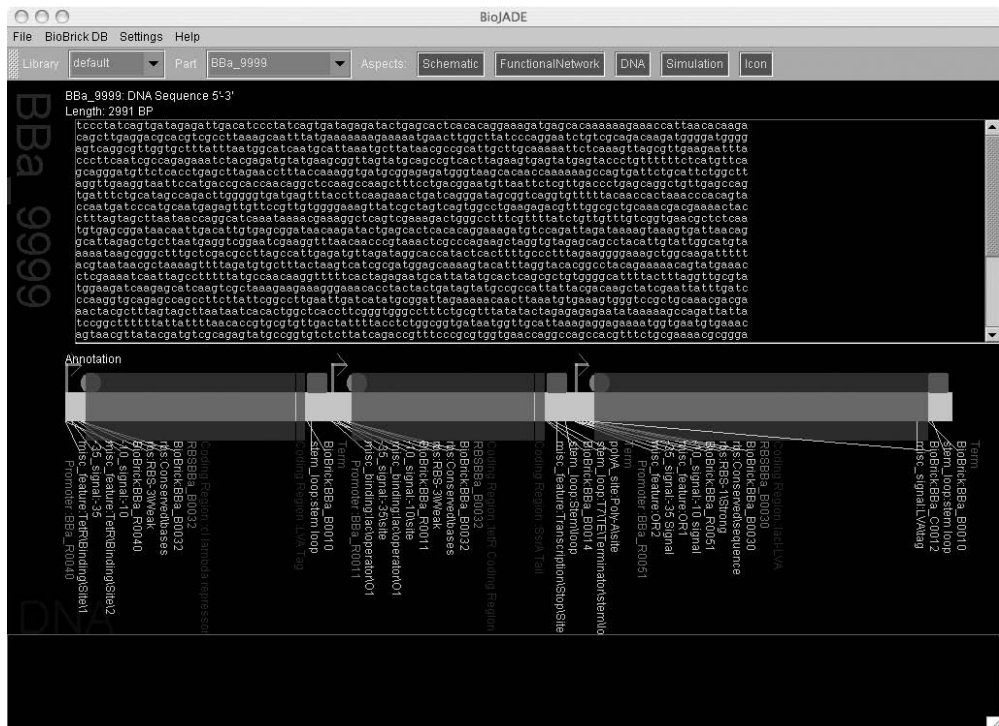


Figure 10.9 The final “output” of BioJADE’s compilation of the Repressilator design.

10.6 Simulations

Since any nontrivial design would also be nontrivial to build in the wet lab, again taking $2\log_2 n$ days to build, having an informative simulation would be a very valuable tool, to test both the system itself and the sensitivity of its various parts to perturbations or differences in parameters. BioJADE thus far integrates two simulators in its core package: Stochasticirator and Tabasco. BioJADE also utilizes a protocol called D-FLUX to run simulations in a distributed fashion.

10.6.1 D-FLUX

BioJADE utilizes the Distributed, FLeXible and User eXtensible (D-FLUX) protocol for simulations. D-FLUX enables the implementation of a series of wrappers that encapsulate essentially any simulator. Simulators such as Tabasco [16] are written in Java and can be directly called by D-FLUX. However, it is also possible to execute any script or executable program on the host computer; thus D-FLUX can run any simulator that is commercially or freely available. D-FLUX also enables the storage of simulation results back into the BioBricks repository database. So, once begun, simulations will return their results not to the original requestor, but to the database itself, permitting persistent and a more widely accessible storage of results.

10.6.2 Stochastirator

Stochastirator is a Gibson-Modified-Gillespie [17, 18] simulator written by Eric Lyons in C++ [19]. Stochastirator is wrapped by writing out the necessary model definitions and parameters, then executing the simulator.

Stochastirator is a stochastic simulator useful for dealing with systems at the single molecule level, rather than the concentration levels often used. Defining a system with BioJADE allows it to be simulated directly with Stochastirator (see Figure 10.10).

The translation program takes in the design and builds interactions between the species (molecules) in the system and the repressors they bind to. The translator also connects the promoter regions to the genes they promote (by wired connections) by simulating the transcription and translation reactions that produce proteins. This process takes into account the cooperative binding of species, an effect that accounts for the nonlinearity.

10.6.3 Tabasco

Tabasco, written by Sriram Kosuri and Jason Kelly, was originally designed to simulate the behavior of phage DNA entering a cell. The D-FLUX wrapper translates the designer's DNA system into the tabasco format and alters rate constants to simulate normal transcription and control. Tabasco simulates, with single-base resolution, the behavior of polymerases, small molecules, and proteins within the host cell (see Figure 10.11).

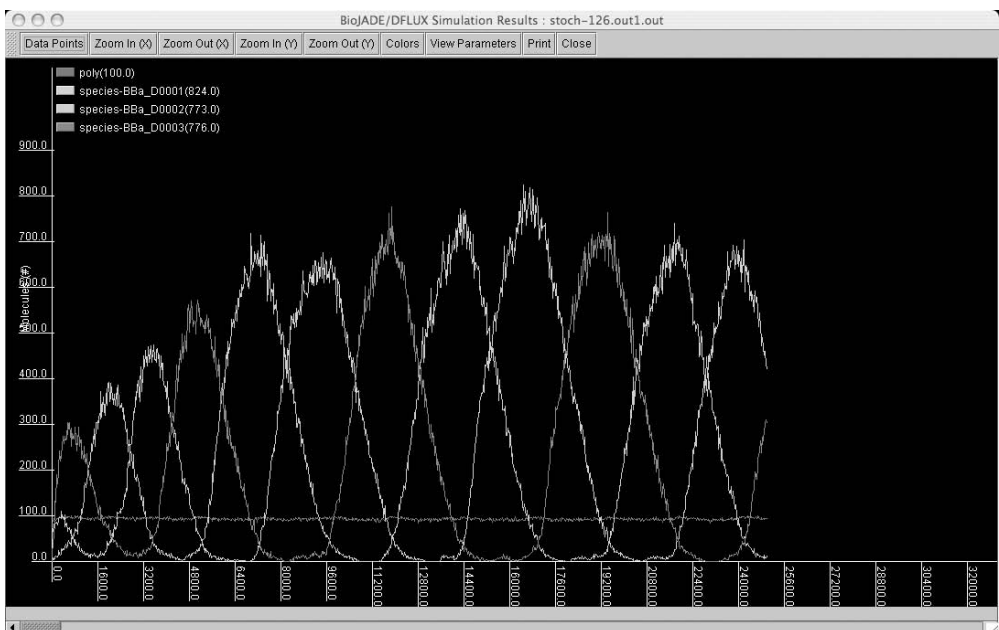


Figure 10.10 A 20-copy plasmid version of the Repressilator, simulated using Stochastirator. Note the noise in the curves. With a lower copy count, that noise overwhelms the oscillatory behavior, and the Repressilator fails to function.

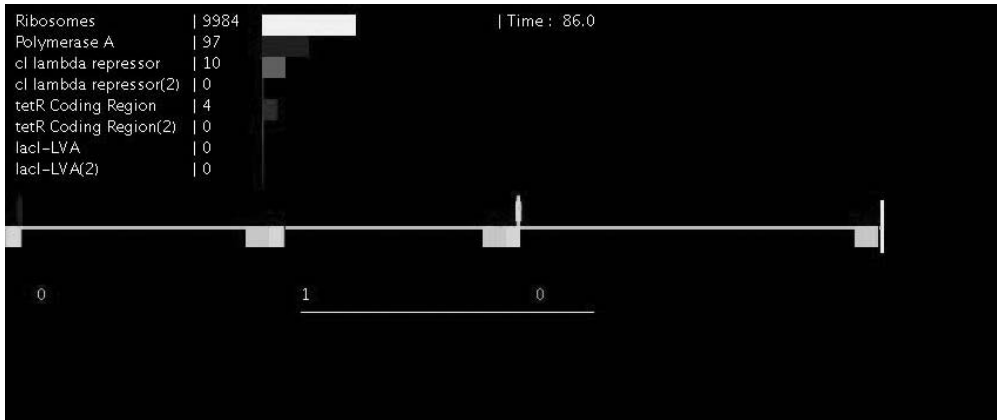


Figure 10.11 Snapshot of the Tabasco simulation of the Repressilator. At this time point, λ repressor is high, which was suppressed by the tetR repressor and is suppressing the production of lacI. The lines below the protein levels show the strand of DNA, and below that the copies of each mRNA. The two ovals above the DNA represent polymerases.

Wrapping Tabasco in D-FLUX was the first practical application of D-FLUX. Writing the wrapper was simple because Tabasco is written in Java. Thus it was possible to hook directly into the native data structures and simply execute the simulator.

Since D-FLUX permits us to execute complicated simulations on (multiple) remote computers, a designer can build a matrix of different values that she wants to test the system with, submit that set of simulations to one or more compute clusters, and proceed to design another system while the remote cluster executes the simulations. She can then check the servers to see if the results are ready. If so, she downloads the datasets and analyzes the data. In the case of Tabasco, this data can be viewed either as plain text values or, more intuitively, as a movie depicting the transcription of each gene. The D-FLUX packaged version of Tabasco automatically generates a quicktime movie out of the simulation data and stores it in the parts repository.

10.6.4 Generating the Simulation

Simulations are generated by taking the entire system as designed and specified, and breaking it back down into its constituent components. Simulations, such as Tabasco and Stochastirator, require explicit interactions to be modeled in their stochastic event-driven model. Thus, every interaction that would take place in the cellular context for each component and set of components must be taken into account. For instance, mRNA transcripts are generated by the interaction of RNA polymerases with the downstream regions from promoters, which generated the RNA polymerases in the first place. Promoters can generate polymerases for attached coding regions but only if they are bound or unbound, depending on the type of promoter system; thus for a stochastic simulation, bound and unbound species are represented as different objects, along with equations governing the interconversion between bound, unbound, and multiply bound states. mRNA inter-

acts with ribosomes, other mRNAs, the same mRNA, small molecules in the system, and cellular degradation mechanisms. The number of potential interactions increases exponentially with the number of system components; thus the simulation can become very large, very fast. Fortunately, not all components interact with each other and thus noninteracting species do not get added to the simulation.

10.7 The Reality Check

There are numerous arguments against the development of biologically encoded circuits and synthetic biological systems in general. Most of these arguments are based on the unreliability or unsuitability of the cell as an engineering substrate. Certainly with the current level of understanding, it is far harder to develop reliable, stable systems in cells. It is less clear that, in the long run, biology will be unsuitable for engineering. That you are reading this chapter is an existence proof that it is possible to build a wildly complicated, reliable biological system that is stable across a fairly wide range of environmental conditions. The fact that bacteria themselves exist is an existence proof of nanotechnology that humans have not been able to produce using silicon, the “preferred” method of building reliable information-processing devices. This section discusses two common arguments against the feasibility of biological circuit design.

10.7.1 Biological Circuit Design Cannot Be as Easy as VLSI Design

Our present understanding of biological materials used as an engineering substrate is about as sophisticated as the understanding of electrical components was in the nineteenth century. The argument is that it is impossible to know the exact behavior of a biological system. There is not yet a good understanding of a part in the actual cell itself, its structure and physical reality. If there were that cell would be nearly impossible to model and simulate effectively to a reasonable accuracy. By analogy, take a twisted, tangled power cable. Calculate, from Maxwell, the exact electromagnetic field through each section of the wire and the resultant current on either end of the line. Performing such a calculation is an exercise in futility; building such a complex “model” eliminates the abstraction barrier we are able to construct in electrical engineering. By the same token, assuming the development of reasonably well insulated parts, with understood load characteristics and external interactions, it is not necessary to understand the entire inner workings of the rest of the cell, just as it is irrelevant to someone using a power cord that a twist in the wire might slightly alter the electromagnetic characteristics of the wire.

10.7.2 Bugs Fight Back

By evolution as it is currently understood, systems made via artificial manipulations to perform certain tasks, such as simply producing a fluorescent protein, experience a greater load on their metabolic capacity. This capacity might otherwise be used in reproduction, thus there is selective pressure against a cell that contains genetic

modifications that increase the load on a cell. This characteristic is both an advantage and disadvantage in synthetic biology.

The disadvantage is obvious: a cell would like to reject load-bearing modifications. A cell that produces red fluorescent protein (RFP) will eventually be overpowered by those that do not. Methods to improve genetic stability of parts include a number of efforts. One well-known and time-honored approach to reducing genetic instability is chromosomal integration, which is useful for many reasons, in particular, the chromosome is never “lost” like plasmids can be. Since chromosomes are also, by definition, low copy (though not necessarily 1, if they reproduce quickly), stochastic effects that are less noticeable with medium to high copy number plasmids (~20–200) become prominent. For example, the Repressilator would never work if encoded in the same manner, but placed on a chromosome rather than a high copy number plasmid, since the noise in the system would quickly out-compete the actual signal. An interesting experiment might be to try to implement the Repressilator as a protein signaling cascade, but implemented on the chromosome, and test how well it works, compared to the genetic Repressilator. Certainly chromosomal integration would limit the engineering space for genetically encoded networks.

Another method of improving genetic stability involves the improvement system performance itself by limiting interactions of the engineered system with the host. This method is the development and engineering of the chasses themselves. These new chasses that are simpler will free more capacity for engineered components, with less interaction with the host.

In addition, chasses can be modified to eliminate recombinases and other factors that help cells to mutate out the parts that are added. George Church’s group at Harvard is working to develop a cell with a totally orthogonal transcription and translation mechanism, such that externally introduced systems will be executed on a virtual machine, much like Java is, interacting with the native system only through well-defined channels.

10.8 Next Steps

10.8.1 Simulations

One of the top priorities in the BioJADE space is in the simulator realm. The top priority in that regard is a more accurate model of RNA creation, interactions, and degradation. The current model in BioJADE follows the simple model of transcription, translation, and slow degradation. New behaviors can be added, such as RNase degradation sites, hairpin/secondary structure cleavage and degradation effects, aptamers (interactions with small molecules), and antisense behavior. This RNA-based simulation will greatly enhance the ability to probe existing systems and new synthetic systems in ways that were not possible before.

RNA-based simulations that can characterize the effect of RNA interactions on the 5’ end of ribosome binding site and coding sequence (RBS-CDS) transcripts will aid in understanding the behavior and limitations of those composite parts. In fact, some failures of modularity of the BioBricks system, uncovered by Endy’s group

while analyzing failures in construction, revealed that such RNA secondary structures (hairpins that sequestered the RBS) were responsible. That this type of interaction could occur is obvious when looking at RNA folding data, but is quite plainly nonobvious if two parts labeled BBa_B0012 and BBa_E0040 are assembled. By simulating and recovering problem structures at the RNA level, it is possible to flag those parts for incompatibility and possibly suggest a workaround (a simple swap of the RBS would eliminate that structure). This integration of simulation data, along with proper annotation of the parts database, will lead to a collection of parts that are well-documented and that work together according to the proper abstraction hierarchy, with exceptions caught by the incompatibility notes in the database.

10.8.2 Parts

As mentioned earlier, the definition of what is a part is a somewhat flexible construct. In the future, as DNA synthesis becomes cheaper and faster than cloning for the construction of large and complex systems, the need for a construction methodology will give way to parts that will be assembled directly, without scars or the need for specific adjustments to suit the construction scheme. As synthesis technology exceeds cloning as the preferred fabrication method, a wider array of parts will become available, without restrictions on the ends or internal restriction sites.

10.8.3 Designing Systems

Ultimately, advances in synthesis and assembly technologies will truly enable the envisioned abstraction barriers. Parts and devices can be designed, with well-defined functions and constraints. Then, the design of complex and large systems through the use of automated tools such as BioJADE will become commonplace. BioJADE has been used in a limited fashion in the Synthetic Biology Summer Design Competition and iGEM (International Genetically Engineered Machines) competitions to aid in simulation and design of some systems. The primary difficulty in using it for this purpose is the lack of well-characterized parts. As more parts are characterized and their features and parameters are entered into the BioBrick database, BioJADE's utility in building these systems will increase.

10.8.4 Measurement

In order to have useful parts, it is vitally necessary to characterize the parts and devices currently in existence as well as those under development. Thus far there has not been a concerted effort to make measurements. In order to fully make use of tools such as BioJADE and the BioBricks registry, parts must be characterized. Figure 10.12 provides an example “data sheet,” much like the old TTL (transistor-transistor logic) data sheets regarding performance characteristics, signal levels, the conditions under which the characteristics were measured, the genetic stability of parts over time, and compatibility data. This data, when stored in the registry, enables automated design tools like BioJADE to make more informed decisions about which parts can be put together, which parts have been tested, and which are reliable.

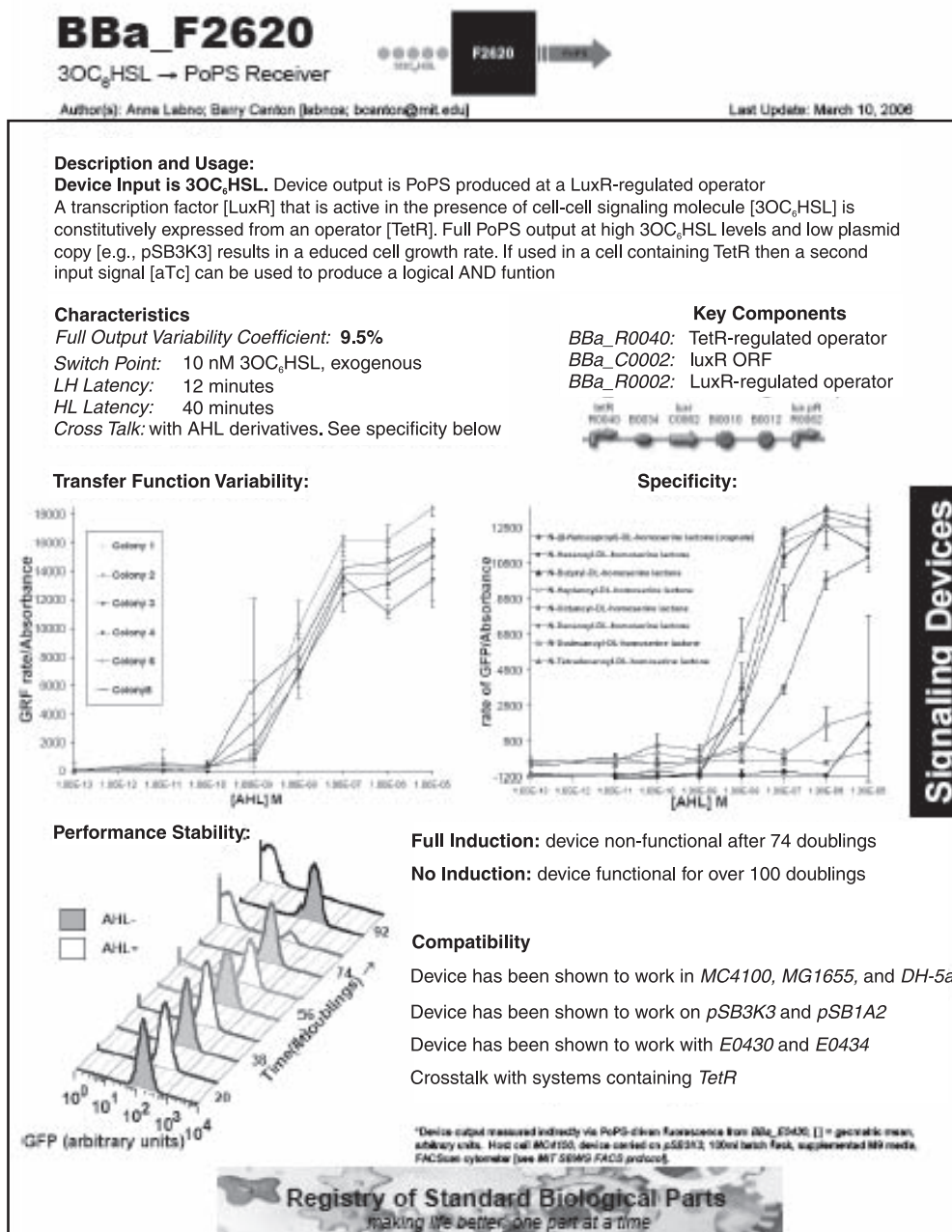


Figure 10.12 A sample data sheet for a signal transducer. The data sheet contains information regarding functional level, variability, specificity, and cross-talk. In addition, the sheet shows performance stability over time, and compatibility. (Image courtesy of The BioBricks Foundation/Barry Canton.)

Currently, the repository contains in excess of 30 terminators, of which 10 have been characterized, exhibiting efficiencies of -109% (acting as a promoter) to 98.4% . There are five characterized ribosome binding sites with two orders of magnitude of relative RiPS performance.

Ultimately, synthetic biological systems will be comprised of myriad components, and the better characterized and tested those components are, the more reliable the devices and systems that are built from them will be. With better-understood components, much of the guesswork and failure modes that are so commonplace in biology will be greatly reduced. Then, the grandiose vision of highly complex, reliable, and effectively designed biological systems will be within reach.

Acknowledgments

Thanks to Randy Rettberg, Drew Endy, Sri Kosuri, Gerald Sussman, the Synthetic Biology Working Group at MIT, the BioBrick foundation, and Registry of Standard Biological Parts. Funding was generously provided by the National Science Foundation.

References

- [1] Kaplan, H. B., and E. P. Greenberg, "Overproduction and purification of the luxR gene product: Transcriptional activator of the *Vibrio fischeri* luminescence system," *Proc. Natl. Acad. Sci. USA*, Vol. 84, No. 19, 1987, pp. 6639–6643.
- [2] Fuqua, W. C., et al., "Quorum sensing in bacteria: The LuxR-LuxI family of cell density-responsive transcriptional regulators," *J. Bacteriol.*, Vol. 176, No. 2, 1994, pp. 269–275.
- [3] Fuqua, W., Winans, S. C., Greenberg, E. P., "Census and consensus in bacterial ecosystems: the LuxR-LuxI family of quorum-sensing transcriptional regulators," *Annu. Rev. Microbiol.*, Vol. 50, 1996, pp. 727–751.
- [4] Greenberg, E. P., "Quorum sensing in gram-negative bacteria," *ASM News*, Vol. 63, No. 371, 1997, pp. 1024–1027.
- [5] Weiss, R., *Cellular Computation and Communications Using Engineered Genetic Regulatory Networks*, Cambridge, MA: MIT, 1992.
- [6] Yang, W., et al., "Rational design of a calcium-binding protein," *Protein Sci.*, Vol. 8, 1999, pp. 2186–2193.
- [7] Looger, L. L., et al., "Computational design of receptor and sensor proteins with novel functions," *Nature*, 423, No. 6936, 2003, pp. 185–190.
- [8] Elowitz, M. B., and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, Vol. 403, No. 6767, 2000, pp. 335–338.
- [9] Vilar, J. M. G., et al., "Mechanisms of noise-resistance in genetic oscillators," *Proc. Natl. Acad. Sci. USA*, Vol. 99, No. 9, 2002, pp. 5988–5992.
- [10] Vilar, J. M. G., et al., "Modeling network dynamics: the lac operon, a case study," Arxiv preprint q-bio.MN/0411036, 2004.
- [11] Knight, T. F. J., "Idempotent vector design for standard assembly of biobricks," *MIT AI Lab Technical Reports*, Cambridge, MA: MIT, 2002.
- [12] Rettberg, R., "Triple-antibiotic (3A) assembly process," *MIT AI Lab Technical Report*, Synthetic Biology Working Group, 2003.
- [13] Knight, T. F. J., "Biobricks standard distribution," *MIT AI Lab Technical Reports*, Cambridge, MA: MIT, 1999.

- [14] Knight, T. F. J., "Biocomp plasmid distribution 1.00 of standard biobrick components," *MIT AI Lab Technical Reports*, Cambridge, MA: MIT, 2002.
- [15] Terman, C., JADE, MIT, *Proc. IEEE*, Vol. 88, No. 1, 2000.
- [16] Kelly, J., S. Kosuri, and D. Endy, "The Tabasco simulation system," *Synthetic Biology Working Group*, Cambridge, MA: MIT, 2001.
- [17] Gillespie, D. T., "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *J. Comput. Phys.*, Vol. 22, No. 4, 1976, pp. 403–434.
- [18] Gibson, M. A., and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *J. Phys. Chem. A*, Vol. 104, No. 9, 2000, pp. 1876–1889.
- [19] Lyons, E., "Stochastirator," computer software, Molecular Sciences Institute, Berkeley, CA, 2000, opnsr.bio.molsci.org/stochastirator/stoch-main.html.

Applied Cellular Engineering

Brian M. Baynes and William J. Blake

11.1 Introduction

In their struggle for survival, biological systems produce a broad range of molecules, including nucleic acids, proteins, lipids, carbohydrates, and other small molecules, from environmentally available precursors. These natural products have a range of biological functions, and many find utility outside of their original synthetic hosts. As cells propagate naturally and can be grown in large vessels, the possibility of synthesizing large quantities of natural products of interest via a “cell factory” is quite enticing. Such syntheses can be largely self-contained, can consume renewable starting materials, and may boast decreased waste generation and smaller environmental footprints than traditional chemical processes. However, cell factory applications are complicated by low molecular yields, resulting in higher production costs.

Overcoming such limitations is one of the central goals of cellular engineering. This discipline and the related field of metabolic engineering have led to the development of cell lines that manufacture large quantities of important molecules such as pharmaceuticals, biodegradable polymers, specialty chemicals, and fuels. As research in these areas advances and competing technologies based on petrochemical feedstocks become more costly, it is widely believed that a significant portion of the trillion-dollar industrial and specialty chemical market will employ biological systems.

This chapter explores the challenges, successes, and future directions in the field of cellular engineering as applied to the biosynthesis of industrially useful molecules.

11.1.1 Biological Systems Engineering

A biological system is analogous to a complex circuit, involving specialized interactions between proteins, DNA, RNA, and small molecules. These circuits are modular in nature, with separable component parts exhibiting distinct functional properties [1–3]. Biological system modularity extends to all levels of complexity, from individual genetic elements and protein domains with specific functional properties, to recurring architectures, or motifs, in complex gene regulatory networks [4–6]. As in traditional systems engineering, biological pathways can be broken down into a series of simpler “parts” that act in concert to control biological function. This enables the dissection of complex cellular pathways into manageable

subsystems that are more easily studied and manipulated. This approach has been used to develop a basic understanding of biological design principles and to develop tools that utilize these principles for cellular and metabolic engineering.

A simple biological module consists of a promoter, the gene or genes expressed from that promoter, and the regulatory proteins (and their associated DNA binding sites) that affect expression of the gene(s). Biological modules are similar to basic circuit elements [7] in that they have distinct inputs, such as regulatory proteins or small molecules, that affect gene expression output (generally measured as the amount of protein produced). Specific sequences of DNA within or near promoter elements serve as binding sites for input factors that can either increase or decrease the probability that the gene is transcribed into mRNA. A basic genetic module is illustrated in Figure 11.1. Gene regulatory networks composed of these modules perform complex computations based on molecular and environmental inputs to produce an output signal that translates into the best mode of survival in a given environment. Computation occurs in the context of network architectures familiar to many engineers, such as negative feedback [8], positive feedback [9, 10], and feed-forward motifs [11]. These discrete motifs are common in biological systems and provide a framework for understanding how the components of a particular pathway interact to control pathway output. Due to the recurrence of these common motifs across many regulatory networks, understanding their input/output relationships may aid in constraining particular designs for pathway engineering.

The engineering of biological pathways is enabled by component and pathway modularity. Regulatory proteins themselves can be broken down into functional domains that can be combined to produce hybrid proteins of novel function [12–14]. Similarly, their cognate binding sites can be inserted into promoter elements, addressing desired regulatory proteins for novel transcriptional control. These discoveries have enabled more complex strategies for regulating gene expression [15].

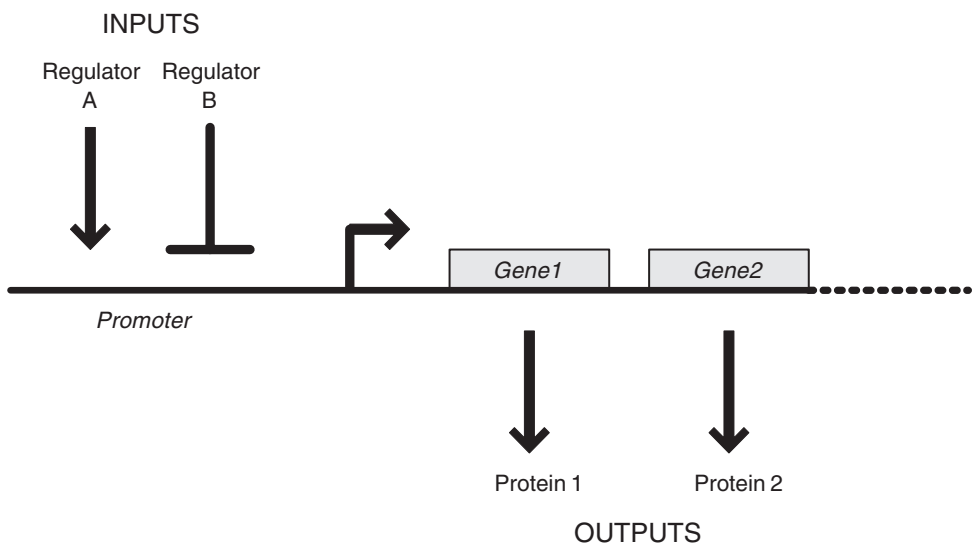


Figure 11.1 A basic gene regulatory module. Regulatory inputs either increase (a) or decrease (b) the expression of one or more genes from a promoter element. Module output is commonly measured as the amount of protein produced.

However, such rational design strategies can be limited by the extent to which we understand functional details of individual component parts. This lack of understanding has prompted an approach that aims to sample a large regime of DNA and amino acid sequence space while directing outcomes based on desired characteristics. Such directed evolution approaches have proven powerful and have led to the creation of more potent compounds and more functional enzymes [16–18].

A modular approach to understanding gene regulation involves the breakdown of complex pathways into a few simple interactions or regulatory events. Such approaches have broadened our understanding of biological design principles, enabling biologists and bioengineers to develop more sophisticated tools for manipulating biological systems. While cellular engineering has the potential to impact global regulatory networks on a genomic scale, initial work has focused primarily on the catalytic machinery of individual pathways.

11.1.2 Cellular Catalytic Machinery

Unlike most laboratory chemical syntheses, where wide ranges of temperature, pressure, solvent, and other conditions can be used to affect selectivity and yield of chemical reactions, biosyntheses are highly constrained by environmental conditions. Cells overcome this limitation through the use of catalytic proteins, called enzymes, which promote chemical transformations with exquisite specificity. Some enzymes enhance the rates of individual chemical reactions by 10^{12} -fold or greater and have evolved to do so effectively even in the complex intracellular milieu. Enzyme activity is generally regulated to carry out specific transformations at different rates in different phases of the cell cycle and in response to environmental cues.

Multistep transformations can be carried out via the action of multiple enzymes in sequence. For example, in the glycolysis pathway, a series of nine enzymes convert intracellular glucose ($C_6H_{12}O_6$) into pyruvate ($C_3H_3O_3^-$) via eight intermediate compounds. To produce the large number of distinct chemical species necessary for function of the cell as a whole, a far larger number of distinct enzymes is required. In *Escherichia coli*, more than 600 enzymes that catalyze a network of almost 750 reactions have been identified [19]. In higher organisms, reaction networks are larger and even more complex.

11.1.3 Early Engineering Successes

The advent of recombinant DNA techniques in the 1970s enabled one to make changes to the genetic makeup of a host and therefore to alter the host's biosynthetic machinery. With some knowledge of the enzymatic pathways and regulatory mechanisms to be adjusted, it is possible to use techniques such as introduction of a plasmid bearing an enzyme-coding gene to a host cell line, knocking genes into or out of a host chromosome, and site-directed mutagenesis [20] to modify the biosynthetic machinery of host cell lines in a rational and directed manner. Initial successes along these lines involved transformation of a plasmid containing heterologous enzyme genes into a new host. The target host was generally one that was easy to manipulate, such as *E. coli*. Expression of the heterologous enzyme genes in this host conferred upon the host new biosynthetic capabilities.

In 1983, Schell at Genex Corp. cloned a cluster of naphthalene degradation genes from *Pseudomonas putida* into *E. coli* and showed that the products of these genes were active and capable of degrading naphthalene introduced into the cell by transport from the media [21]. Around the same time, scientists at Amgen cloned the gene for naphthalene dioxygenase into *E. coli* in a different vector context [22] and observed varying degrees of a blue-purple product under different growth conditions. They suggested that this enzyme acts on intracellular indole, a natural degradation product of the amino acid tryptophan, to produce an intermediate that is spontaneously converted to indigo.

This work spurred an interest in heterologous enzyme expression for the production of useful chemicals. Later that decade, Lonnie Ingram's group at the University of Florida combined two previously cloned genes, pyruvate decarboxylase and alcohol dehydrogenase II from *Zymomonas mobilis*, into an artificial operon they designated *pet* (ethanol production) and transformed *E. coli* with it [23]. These heterologous enzymes, which convert the metabolite pyruvate into ethanol, shifted the primary fermentation products of the transformed strain from lactate and acetate to ethanol.

Similarly, in studies of the biology of *Alcaligenes eutrophus*, Peoples and Sinskey at MIT used genetic complementation to identify *phbC*, a polymerase responsible for biosynthesis of polyhydroxybutyrate (PHB) from 3-hydroxybutyryl-CoA [24]. They subsequently cloned the gene for this enzyme along with the genes for *phbA* (a ketothiolase) and *phbB* (a reductase) into *E. coli*. Acting in series, these three enzymes allow *E. coli* to convert intracellular acetyl-CoA into PHB. This led to the founding of Metabolix, Inc. (Cambridge, MA), which today is commercializing biosynthetic methods for manufacture of polyhydroxyalkanoate (PHA) plastics.

The successes of these and other early efforts in the field led to the formal definition of metabolic engineering as a discipline by James Bailey in 1991 [25]. Since then, several reviews of cellular and metabolic engineering, a textbook, and a dedicated journal (Elsevier's *Metabolic Engineering*) have appeared [26–32].

11.2 Engineering Tools

Modern cellular engineering projects use a range of computational and experimental techniques to achieve an optimization objective. Typical projects involve several steps in a cycle, including (1) development and/or refinement of a system model, (2) use of the model to generate a hypothesis, (3) development and execution of an experimental plan to test the hypothesis, and (4) analysis of the experimental results. In this section, tools available at each of these stages and examples of their use are overviewed.

11.2.1 Network Models and Analysis

Because of the inherent complexity of biological systems, computational models are used to predict system performance. For chemical transformation objectives, the starting place for such computational work is a model of all the chemical reactions that a cell may carry out. In its simplest form, this model may only indicate directional connectivity (that it is, for example, feasible to synthesize a product from a

reactant in the presence of an enzyme) or may contain thermodynamic or kinetic information. These models allow the cellular engineer to perform several types of network-wide calculations, including assessing feasibility of synthesis of a metabolite from a precursor given a set of enzymes, analysis of global metabolic flux, and assessment of optimal metabolic network perturbations.

11.2.1.1 Identification of Biosynthetic Pathways

Computational tools have been developed to identify possible enzymatic pathways that will convert a specified precursor molecule to a desired product. Algorithms of this type require a database of available reactions and simulate these reactions computationally. In one formulation, available enzymatic transformations are applied to a metabolite pool in successive rounds to simulate reactions in series [33]. After a certain number of iterations or other constraints on the computational complexity are reached, the algorithm stops, and paths leading to the target molecule (if any) are reported. An alternate algorithm that deals with certain types of reactions more efficiently has also been developed [34]. Recently, a strategy that first found use in the petrochemical industry has been extended to postulate formation of entirely new biosynthetic compounds given a set of reaction rules for molecular classes, rather than distinct chemical species [35].

After reaction paths of interest have been determined, the set of reactions in each path can then be stoichiometrically balanced and summed to yield a net chemical reaction for the path. Generally, the stoichiometry of the product, the other reactants and products involved, and the cofactors involved will be different as a function of path. The relative merits of particular paths can then be compared on the basis of these net reactions. Other important pathway-dependent factors that are not apparent from the stoichiometry include differences in flux regulation and flux limitation due to competition for cofactors.

11.2.1.2 Flux Analysis and Perturbation Strategies

In addition to questions of feasibility, network models allow engineers to probe how network changes will affect metabolic flux rates. In the case of metabolite overproduction, one would like to increase the flux of material from an available precursor toward the metabolite of interest without disturbing the remainder of metabolism, which presumably has been optimized by evolution of the host.

Basic models of flux have been developed that capture the key independent variables that can be used to alter metabolic flux. The two fundamental flux processes that must be considered are:

1. Transport of molecules across domain boundaries, such as the cell membrane or mitochondrial membrane, and
2. Enzymatic chemical reactions that transform one or more molecular species into different molecular species.

Membrane transport can be thought of as a three-step process of diffusion or active transport of a molecule to, through, and away from the membrane. In general, the

first and last of these can be assumed to be at equilibrium, and the actual process of transport through the membrane is rate limiting. The rate of transport can then be modeled as

$$v_i = k_{m,i}a(C_i^A - K_iC_i^B) \quad 11.1$$

where v_i is the reaction flux of species i (in mol/s) from domain A to B, $k_{m,i}$ is the mass transfer coefficient (in m/s), a is the interfacial area (in m²), C_i^A and C_i^B (in M) are the concentrations of the molecule being transported in domains A and B respectively, and K_i is the ratio of the concentration in domains A and B at equilibrium.

If a required precursor molecule enters the cell via passive transport, the easiest way to increase the membrane flux is to increase the concentration of this molecule in the media. In active transport, where a protein or other molecule is responsible for transport, the mass transfer coefficient can also be altered by altering the concentration of the transporter in the membrane or its efficiency.

Enzymatic chemical reactions are diverse and often multistep. In the simplest case, an enzymatically catalyzed unimolecular reaction, the enzyme and reactant bind each other, the reactant is converted to a product, and finally the product is released from the enzyme. In general, the concentration of the enzyme is far less than the concentration of the reactant, and the intermediate enzyme-substrate complex can be assumed to be at steady state, leading to the Michaelis-Menten model for an enzyme-catalyzed reaction rate:

$$v = -\frac{dC_S}{dt} = \frac{dC_P}{dt} = \frac{k_{cat}C_EC_S}{K_m + C_S} \quad 11.2$$

where v is the reaction flux (in M/s), C_S is the substrate concentration (in M), C_P is the product concentration (in M), k_{cat} is the catalyst turnover number (in s⁻¹), C_E is the enzyme concentration (in M), and K_m is the Michaelis constant (in M). An important consequence of this rate relationship is that the reaction velocity is proportional to the enzyme concentration and turnover number. In many cases, the substrate concentration will be lower than the Michaelis constant ($C_S \ll K_m$), and the reaction flux will be proportional to substrate concentration and a lumped parameter representing the enzyme “activity” equal to $k_{cat}C_E/K_m$.

Consider the reaction scheme shown in Figure 11.2 for converting precursor molecule A into intracellular metabolites. Here, environmental precursor molecule A is transported into the cell and converted into either B or C via enzymatic reactions.

Assuming linear relationships between flux and transporter or enzyme activity, the metabolic system can be represented by the following system of equations:

$$\begin{aligned} \frac{dC_{Ao}}{dt} &= -k_1C_{Ao} \\ \frac{dC_{Ai}}{dt} &= k_1C_{Ao} - (k_2 + k_3)C_{Ai} \\ \frac{dC_{Bi}}{dt} &= k_2C_{Ai} \\ \frac{dC_{Ci}}{dt} &= k_3C_{Ai} \end{aligned} \quad 11.3$$

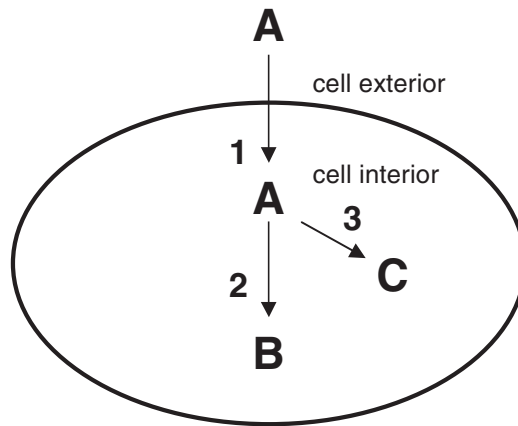


Figure 11.2 A simplified metabolite network. Environmental precursor molecule A moves into the cell via transport process 1 and is converted into metabolites B and C via enzymatic reactions 2 and 3.

where C_{Ao} , C_{Ai} , C_{Bi} , and C_{Ci} are the concentrations of A outside the cell, A inside the cell, B inside the cell, and C inside the cell respectively; k_1 is the rate constant for transport of A into the cell, and k_2 and k_3 are the rate constants for intracellular A to B and C respectively.

If we desire to overproduce metabolite B, we need to increase the mass flux from A to B. The simplest genetic change that could be attempted to increase the flux to product B is to increase the activity of the enzyme catalyzing the reaction of A to B, hence increasing k_2 and dC_{Bi}/dt . Alternative single enzyme manipulations that could be beneficial are increasing k_1 and decreasing k_3 .

Activity modulations of this type can be attempted by either altering the specific activity of the enzyme (k_{cat} and/or K_m) and/or by altering the enzyme's concentration (C_E). Some experimental methods of performing these modifications, including altering the gene's promoter, mRNA half-life, gene copy number, codon usage, and specific amino acid sequence of the protein, are addressed later in this chapter.

11.2.1.3 Consequences of Perturbation

Strategies of this nature, while a reasonable starting place, have met with limited success in practice. This type of approach generally shifts metabolism away from its evolutionary optimum and places the engineered cell at a growth or robustness disadvantage. This can be due to accumulation of toxic intermediates or via virtual starvation as large amounts of resources are diverted toward metabolite overproduction. Another challenge is that metabolic networks in general do not have a single "rate limiting step" and therefore perturbations at many steps of a metabolic transformation may be required to bring about a significant change in output [36].

The host's regulatory network may also resist an engineering change. One general reason for this is that enzyme activities are often regulated by the presence of the substrates, products, or other intracellular species they affect in such a way that flux and intracellular concentrations are stabilized. Common regulatory mechanisms include [37]:

- Feedback repression of enzyme synthesis and/or enzyme activity by a product metabolite
- Feedback promotion of a competing enzyme activity by a product metabolite

Some simple regulatory strategies are shown in Figure 11.3. In effect, the activity and transport constants (k_1 , k_2 , and k_3) are not independent of the metabolite concentrations, and the signs of these effects, which are evolved to enhance stability of the overall system, generally counteract external changes. Regulation of this type is a significant complication in metabolic engineering [38].

To avoid regulatory consequences to a metabolic engineering change, the engineering change should disturb metabolism to the smallest extent possible. Ideally, only fluxes to a given product will be changed, and none of the concentrations of the intermediate metabolites will be changed. If this can be accomplished, there should be no regulatory consequences to the change. This has the further advantage that the resulting system will be as close as possible to the unengineered system, which is presumably at an “evolutionary optimum” for growth and propagation.

In the context of the metabolic network in Figure 11.2, the increase in k_2 proposed in the previous section to overproduce metabolite B would have the secondary effect of decreasing the C_{Ai} from its evolutionary optimum, assuming neither the rate of influx of A into the cell nor the rate of conversion of A to C changes. Therefore, one would also like to increase k_1 , increasing the rate of transport of A into the cell, to compensate for the increased flux to product B. If properly balanced, these changes can increase the flux toward B from extracellular A without altering the remainder of cellular metabolism. This concept was proposed by Kacser and Acerenza [39] and resulted in a set of rules for the enzyme concentration changes required to bring about a desired flux perturbation objective. These rules highlight the fact that it is unlikely to be optimal to make single changes in isolation or to boundlessly overproduce every enzyme in a particular biosynthetic

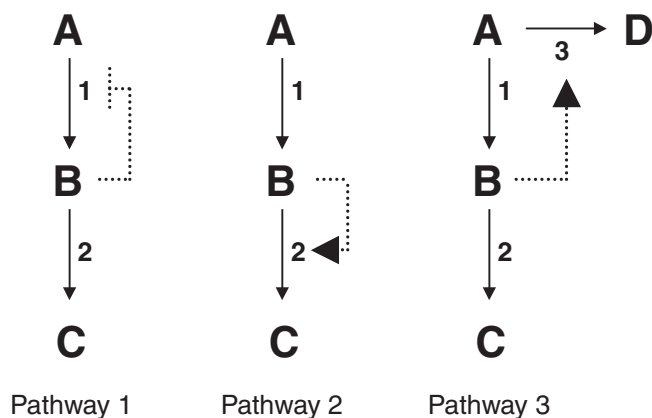


Figure 11.3 Three strategies for regulation of the concentration of metabolite B are shown. Regulatory interactions from a metabolite to an enzyme are shown as dotted lines. In Pathway 1, the presence of B decreases the activity of enzyme 1, decreasing the flux from A to B. In Pathway 2, the presence of B increases the activity of enzyme 2, increasing the flux from B to C. In Pathway 3, the presence of B increases the activity of enzyme 3, which diverts flux from the branch containing B and C to the branch containing D.

pathway. They also assume that the machinery required to overproduce the pathway enzymes themselves does not have any effect on the rest of the system and that the overproduced product (if remaining in the cell) has no effect on the system. In practice, this type of differential flux balance is difficult to achieve. The activities of each perturbed enzyme in the pathway must be precisely balanced so they do not alter the intermediate metabolite concentrations. Genetic tools do not currently offer the metabolic engineer this level of precision. Developing strategies to overcome this limitation is an active line of research.

11.2.1.4 Steady-State Analysis

For many metabolic systems, the cellular expression and regulation functions on a much shorter time scale than the cellular lifetime and fermentation process lifetime. In such cases, metabolic fluxes can be assumed to result in a pseudo-steady-state system where the time derivatives of metabolite concentrations that are not edge nodes (not singly connected) are zero. In general, this can be written as $S\mathbf{v}=\mathbf{b}$, where S is a matrix of stoichiometric coefficients connecting the fluxes to each other at nodes, $\mathbf{v}$ is a vector of fluxes, and $\mathbf{b}$ is a vector of edge fluxes [40, 41]. In the case of the system in equation (11.3), this would be represented via a single equation $0 = \nu_1 - \nu_2 - \nu_3$, expressing the fact that the net flux of A in the cell is zero.

Such systems are almost always underdetermined because there are usually more fluxes than nodes and there are mass balance relationships between the fluxes. These systems can, however, be used as constraints in optimizing an objective function, such as maximizing amount of biomass synthesized [40]. The ability to predict flux distribution in this manner allows generation of hypotheses about how metabolic network perturbations will affect a secondary objective, such as metabolite overproduction. Proposed genetic alterations, such as gene additions and deletions, can be tested *in silico* by adjusting the stoichiometric matrix, recalculating the optimal fluxes, and ascertaining whether the new scenario results in greater metabolic flux toward the target metabolite. When performing addition to or deletion of genes from the host, often absolute growth rate is no longer the appropriate objective function, as the modified cell has not had time to evolve to maximize its growth rate with its new genotype. In such cases, prediction of performance with a minimal flux disturbance criterion [42] can yield superior results.

Stoichiometric models and flux predictions can be further refined by additional constraints from experimental data [43]. Measurement of steady-state concentration of many important metabolites is possible via mass spectrometry and other techniques. In addition, isotopic labeling of precursor molecules provides additional information on the flux distribution of particular atoms [44].

11.2.2 Experimental Methods

The previous sections provide a theoretical framework for the rational design of biological pathways to optimize production of native or heterologous products. The inaccuracies of current metabolic network models limit the predictions of such models to be primarily directional, in that quantitative predictions are rarely validated *in vivo*. Quantitative models may predict, for example, that increasing

expression of a particular enzyme will enhance production of a desired molecule produced in a reaction catalyzed by the enzyme. These simple changes in metabolic or cellular pathways rarely yield the desired result due to unanticipated pleiotropic effects, or the buildup of toxic intermediates. As a result, biological engineering often requires a combination of rational design and experimental trial and error. This is a feedback process where initial models guide experimental direction, and experimental data are used to further refine models that can be applied to more complex systems. Critical to this process is the development of tools that enable tunable modification of target gene expression to both identify pathway function and, ultimately, rewire pathways for a desired goal.

A central challenge for cellular and metabolic engineering is the development of techniques and tools for the precise control of gene expression. The following examples demonstrate various methods for controlling gene expression and focus particular attention on experimental systems designed to control the expression of genes involved in the production of a desired product or in the detection of a pre-defined molecule. While these methods cover a broad range of regulatory strategies, they point toward the challenges and potential benefits of engineering cell function.

11.2.2.1 Transcriptional Control Strategies

A primary mechanism for simple transcriptional control involves use of inducible promoter systems. Native promoters that sense and respond to extracellular molecules have been used, and in some cases engineered, to quantitatively control the level of a particular protein product. Such systems commonly include components from the lactose and arabinose operons of *E. coli*, and the tetracycline resistance genes of transposon Tn10 [45]. The *lac* operon is a paradigm of bacterial gene regulation that has been studied for over four decades [46], while components of the arabinose operon and tetracycline repressor-operator system have been characterized extensively [47–49]. The modularity of these well-studied components, illustrated in Figure 11.4, has led to their wide use in a variety of organisms ranging in complexity from bacteria to mammalian cells. In fact, several commercially available “rheostat-like” expression systems, including Invitrogen’s pBAD, Strategene’s LacSwitch, and Clontech’s Tet-On/Off systems, use these components for control of heterologous gene expression in bacterial and mammalian hosts.

Despite the utility of inducible promoter-based systems, their precise expression levels can be difficult to control over a broad range, and addition of specific inducer molecules can be costly and inconvenient. In addition, there are a limited number of well-characterized inducible promoter systems available, making the development of alternative approaches desirable. Alper et al. [50] took a unique approach to modulating transcription in *E. coli* that used a combination of random promoter mutagenesis and expression strength screening. The aim of this approach was to develop a library of promoters with different constitutive expression strengths to more carefully tune engineered expression systems. Briefly, a constitutive *E. coli* promoter was randomly mutated using error-prone PCR to create a library of promoters exhibiting a wide range of expression strengths. A single round of mutagenesis and screening produced promoters with expression efficiencies that linearly

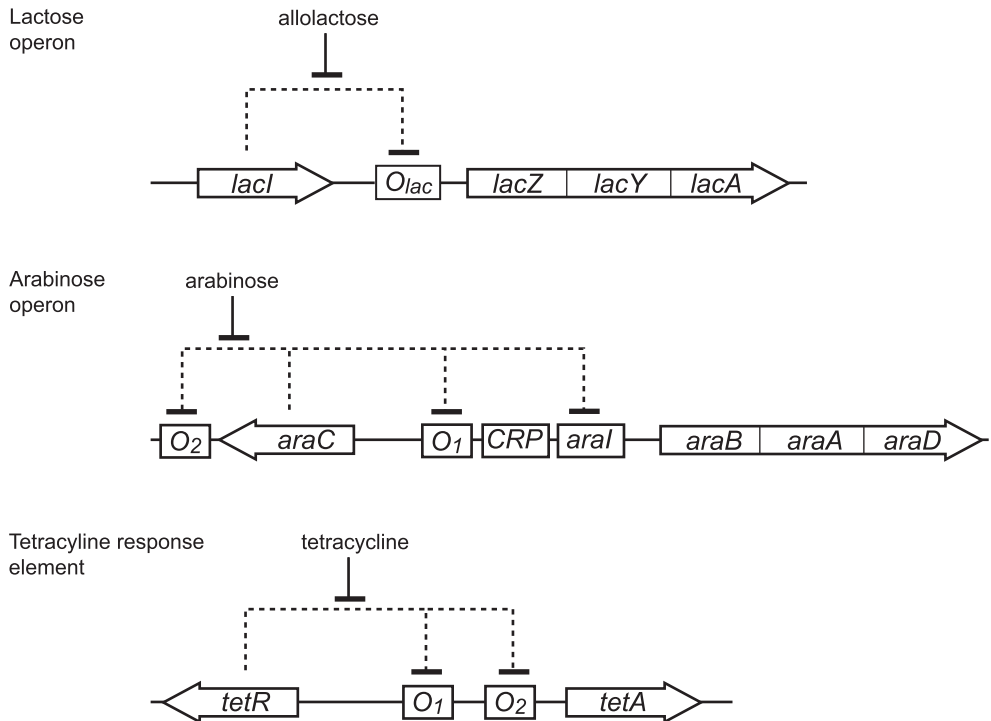


Figure 11.4 Example of inducible promoter systems. Genes from the lactose operon, arabinose operon, and tetracycline responsive element are identified by open arrows, indicating direction of transcription. Regulatory elements are shown as open boxes, and regulatory interactions are indicated by broken lines. Binding of the *lac* repressor (*lacI*) to the *lac* operator (*O_{lac}*) is inhibited by allolactose, while binding of the *tet* repressor (*tetR*) to operators *O₁* and *O₂* is inhibited by tetracycline. Regulation of the arabinose operon is more complex, as arabinose inhibits binding of *araC* to *O₂*, preventing a DNA looping mechanism that negatively regulates operon expression.

spanned approximately two orders of magnitude, highlighting the combinatorial power of mutate-and-screen approaches.

Alper et al. demonstrated the utility of their promoter library by creating a set of *E. coli* strains that differed in the expression of a single gene in the lycopene biosynthesis pathway. Lycopene is a carotenoid antioxidant with a variety of therapeutic benefits, and is produced in *E. coli* through the nonmevalonate isopentenyl diphosphate synthesis pathway. Previous work using inducible promoters in various host strains demonstrated that altering the expression levels of genes in this pathway can enhance production of lycopene [51]. Using their promoter library, Alper et al. demonstrated that expression of *dxs*, a gene in the lycopene biosynthesis pathway, reaches a level that is optimal for lycopene production. Increased expression of *dxs* beyond this optimal level results in a decrease in lycopene production, likely due to suboptimal levels of downstream enzymes in the pathway that may result in increased levels of toxic intermediates. However, when enhanced *dxs* expression was promoted in a host that had been modified to express higher levels of downstream *idi*, *ispF*, and *ispD* genes [52], lycopene production increased linearly with *dxs* expression. These data show that a reaction catalyzed by *dxs* is rate-limiting in the production of lycopene, and demonstrate the utility of the promoter

library in quantitative analysis of the effects of single enzyme levels in a metabolic pathway.

A second approach that directly addresses limitations of the earlier described inducible promoter systems involves the design of custom proteins that sense and respond to particular inputs. Looger et al. [53] combined computational protein design with experimental work to create a set of proteins tailored to bind predefined targets, including trinitrotoluene (TNT) and serotonin. Their rational design approach involved the use of a docking algorithm that varied specific amino acids at the receptor-ligand interface until the global minimum of a semiempirical potential function was identified. This deterministic procedure was applied to five proteins from the *E. coli* periplasmic binding protein family, identifying optimal amino acid substitutions for high-affinity binding to predefined ligands. Seventeen predicted designs were evaluated experimentally by targeted PCR-based mutagenesis of wild-type genes.

All computationally designed proteins bound their specified targets with a range of affinities extending to the nanomolar level, with negligible binding to their original, native ligands. Further, two custom-designed proteins based on the ribose and glucose binding proteins were shown to alter the expression of a gene target in response to binding of TNT. Upon binding to their natural ligands, native versions of these proteins initiate a signal transduction cascade ending in OmpR upregulation of the ompC promoter. By expressing β -galactosidase from the ompC promoter, Looger et al. showed that extracellular TNT caused up-regulation of target gene expression, demonstrating the feasibility of custom designing receptors that mediate target gene expression upon binding to a ligand.

Similar methods for altering gene expression also rely on custom-made DNA-binding molecules. These include the use of Zinc Finger DNA-binding domains fused to transcriptional repressors or activators, triplex-forming polynucleotides, and synthetic polyamides—all of which have been used to regulate gene transcription [54]. It is interesting to consider the use of these techniques to customize protein mediators that would act in engineered metabolic pathways to sense and respond to the presence or absence of particular pathway inputs, intermediates, or outputs to enhance flux through the pathway. Such a technology will have utility in probing biological systems to determine the key genes, proteins, and interactions that determine system function, but will also play an important role in modifying such systems to exploit cell function.

11.2.2.2 Post-Transcriptional Control Strategies

Gene expression is controlled to a large extent by regulation at the transcriptional level, leading to the development of various strategies for engineered transcriptional control. There are, however, compelling reasons to develop new strategies for controlling intracellular protein levels based on post-transcriptional control mechanisms. Post-transcriptional control can be combined with transcriptional control to enable more precise regulation of protein levels. More importantly, post-transcriptional control enables the additional flexibility of tuning expression levels in a manner that is independent of the promoter being used, allowing use of native promoter elements. Various post-transcriptional control mechanisms have been de-

veloped to control gene expression, and several of these have been used to engineer metabolic pathways for more efficient production of desired outputs.

One method for post-transcriptional control involves the modulation of messenger RNA (mRNA) stability based on the addition of stabilizing sequences that can alter the level of protein output from a single mRNA transcript [55]. Work has shown that mRNA can be less stable under conditions where cellular resources are strained—for example, in the overproduction of a heterologous protein [56]. By introducing stabilizing stem-loop sequences within the untranslated region (UTR) between the transcription and translation start sites of a particular gene, the half-life of the associated mRNA transcript can be increased. This results in a greater protein yield from each individual transcript. This method was employed to create a library of cassettes that could be inserted within the 5' UTR of any *E. coli* gene and was shown to modulate protein levels over a broad range [57].

In bacterial regulatory networks, it is common for genes involved in a particular metabolic or cellular pathway to be coordinately expressed from a single promoter in a genetic element called an operon. All genes in a single operon are transcribed together in response to signals that activate expression. Smolke et al. took advantage of native RNase enzyme specificity and mRNA-stabilizing elements to tune the expression levels of genes contained in a single operon in *E. coli* [58]. Placement of RNase cleavage sites between coding regions of an operon allows for the physical separation of the transcript into its distinct coding regions through RNase-mediated cleavage. This enabled the use of particular UTR sequences—specific to each transcript—to independently control the levels of each coordinately transcribed gene product. This approach was implemented by placing RNase E sites from the *Rhodobacter capsulatus* *puf* operon and *E. coli* *pap* operon between coding regions of a novel operon composed of two reporter genes. Novel secondary structures at the 3' and 5' ends of the transcripts protected against exonuclease cleavage and allowed for altered mRNA stability, leading to independent, tunable expression of each reporter gene transcribed from a shared promoter.

The utility of this method for altering protein output in a cellular pathway was demonstrated by applying it to control the of flux through a carotenoid-producing metabolic pathway in *E. coli* [59]. Two genes involved in the conversion of phytoene to lycopene and lycopene to β -carotene—*crtI* and *crtY* respectively—were coordinately expressed from a single promoter in *E. coli*. Directed placement of RNase E cleavage sites between the coordinately expressed genes, together with varying 5' and 3' UTR sequences that mediated mRNA secondary structure, resulted in widely varying production levels. This was reflected in ratios of β -carotene to lycopene that varied over two orders of magnitude. Analysis of *crtI* and *crtY* production levels showed that balanced intracellular levels of *crtI* and *crtY* protein resulted in higher flux through the pathway. Altering the levels of each of these proteins relative to the other resulted in the buildup of one or more intermediates with decreased flux through the pathway. Smolke et al. were able to demonstrate that tools for post-transcriptional control can be used to alter flux through a metabolic pathway, enabling rational design and optimization of pathway function.

While the work by Smolke et al. demonstrated the utility of post-transcriptional control strategies that involve modulating the stability of mRNA, other methods have been developed that involve the use of native ribozymes [60] or riboswitches

[61] for post-transcriptional control of gene expression. Ribozymes are RNA molecules that catalyze chemical reactions, such as the cleavage of other RNA molecules. Riboswitches are RNA molecules that regulate their own translation through binding of small molecules or metabolites. The use of these tools in controlling gene expression will likely find more widespread use as the number of riboswitch elements increases and their functionality and ease of use is effectively demonstrated.

11.2.2.3 Translational Control Strategies

While transcriptional and post-transcriptional methods for gene expression control offer remarkable flexibility and utility, there exists another layer of control at the translational level. The previous section described methods that enabled the modulation of mRNA stability and therefore the number of protein molecules produced from a single mRNA transcript. However, there are other methods that take advantage of the translation process itself. Foremost among these involves the alteration of gene codon content, optimizing synonymous codon usage for maximal protein production.

This approach was successfully applied in the development of an engineered pathway to produce terpenoids in *E. coli* [62]. Terpenoids are a broad group of natural compounds useful in a variety of organic syntheses, and are generally isolated from organisms that are difficult to cultivate in large quantities. Martin et al. sought to produce amorphaadiene, a terpenoid precursor to artemisinin, an antimalarial product with high promise in the development of a potent, economical treatment for malaria. Artemisinin is found naturally the plant *Artemisia annua*, and it is difficult to isolate the compound in large quantities. Although the native *E. coli* DXS pathway is available for the production of isoprenoid precursors required for the large-scale production of carotenoids in *E. coli* (gene expression control strategies for optimizing this pathway have been described above), Martin et al. chose to use the mevalonate pathway from *Saccharomyces cerevisiae*, or baker's yeast. The mevalonate-dependent isoprenoid pathway is used by eukaryotes for the biosynthesis of isoprenoids.

The rationale behind transplanting the mevalonate pathway from *S. cerevisiae* into *E. coli* is centered on the difficulty in producing high levels of isoprenoids in *E. coli* through the use of the native DXS pathway. This may be due to pleiotropic effects resulting from the artificial modulation of native *E. coli* genes, an effect that would likely be avoided by using a foreign pathway. The mevalonate pathway was coupled to a plant-derived amorphaadiene synthase gene (ADS) for conversion of farnesyl pyrophosphate (FPP) to amorphaadiene. A critical challenge to this approach centers on the differences in codon usage between organisms. Rare codons can decrease the translational efficiency of an mRNA transcript, resulting in lower protein yields. Prior work has demonstrated that codon optimization, or the replacement of rare codons with more frequently encountered codons for a particular amino acid, will result in increased translational efficiency and higher protein yields [63]. To increase flux through the heterologous isoprenoid pathway in *E. coli*, Martin et al. synthesized a codon-optimized ADS gene for use in *E. coli*. By optimizing codon usage for *E. coli*, an increase in terpene synthesis of approximately two orders of magnitude was achieved. This demonstrates that limitations

at the translational level can be overcome through a rational strategy of codon optimization.

11.3 Case Study: Production of 1,3-Propanediol in *E. coli*

1,3-Propanediol (13PD) is a specialty chemical that is structurally related to its isomer 1,2-propanediol and to glycerol. 13PD, however, is significantly more difficult to manufacture and is consequently much more expensive than related molecules, which has limited its use industrially as a raw material. If a low-cost 13PD synthesis method were available, it would be a molecule of choice in synthesizing a variety of polymer products. This promise has spurred cellular engineers to develop several cell lines for manufacture of 13PD. To bring 13PD biosynthesis costs down to the commodity level, a biosynthetic process must convert a low-cost starting material such as a simple sugar into 13PD in a single fermentation at high yield.

Initial metabolic engineering efforts focused on transferring catalytic machinery from *Klebsiella pneumoniae*, which can ferment glycerol anaerobically and produce 13PD, into *E. coli* [64]. Expression of heterologous dehydratase and oxidoreductase enzymes in *E. coli* grown on glycerol resulted in the formation of 0.5g/L of 13PD, demonstrating that these enzymes could be expressed in an active form in the new host. This level of productivity, however, was far from industrially useful, and the system still required the expensive starting material, glycerol.

Later, DuPont scientists developed *E. coli* strains that could directly ferment glucose and produce 13PD. Initially, the resulting strains produced 13PD at similarly low yields [65]. The yield of 13PD was increased by the addition of genes that reactivated the heterologous dehydratase and those that promoted the conversion of glucose to glycerol—including glycerol-3-phosphate dehydrogenase and glycerol-3-phosphatase—and by the removal of genes that promoted off-pathway and reverse reactions, such as triosephosphate isomerase, glycerol dehydrogenase, and glycerol kinase. Finally, it was observed that *E. coli* possessed a native enzyme capable of producing 13PD from its precursor 3-hydroxypropionaldehyde, and the presence of the heterologous dehydratase that carried out this function in fact resulted in a lower yield of 13PD. These and other changes resulted in a strain that produced 130g/L of 13PD, an improvement of more than two orders of magnitude [66]. Together with Tate and Lyle, Dupont is now commercializing a full-scale process for 13PD biosynthesis. In addition to showing strong economics, the biological route to 13PD avoids use of acrolein and ethylene oxide, both highly toxic starting materials used in conventional chemical synthesis of 13PD.

11.4 Frontiers

Today, the greatest challenge in cellular engineering is reducing the time required to engineer a cell factory for a new manufacturing objective. The cyclic development process overviewed in the preceding section, involving model development, hypothesis generation, experiments, data analysis, and repetition, is expensive and time consuming. New strategies for cutting the number of iterations required and/or

reducing the duration of each stage will increase the range of industrial processes cellular engineering can be applied against, and further increase the rate of development in the field.

The ability to rapidly construct and modify biological systems of interest with emerging DNA synthesis technologies [67–70] promises both to extend the range of hypotheses that can be tested and to accelerate the pace at which experiments can be performed.

Increasing recognition of the modular nature of biosynthesis pathways and exploitation of this architecture where possible will also be valuable. Through creative manipulation of enzymes, heterologous expression of engineered enzymes can be used to produce new molecules not previously known to be available biosynthetically. Recent work in generating novel combinations of modules in polyketide synthase (PKS) genes [71] produced a library of 154 novel PKS genes, and about half as many new polyketides. Coupled with an appropriate large-scale selection methodology, such a combinatorial library technique can rapidly identify a pathway capable of producing a synthetic molecule of interest.

Because metabolism is dynamic, an optimally engineered cell should observe and respond to intracellular conditions in a controlled fashion as conditions change [72]. To make this a reality, new methods for engineering regulatory networks and elements of these networks such as promoters and intracellular sensors are required.

11.5 Conclusion

Today, cellular engineering is a robust and rapidly developing field. This trend will continue with the aid of innovative research and increasing price pressures on competing petrochemical-based processes. The emerging field of synthetic biology will play a large role in the continued development of cellular engineering by providing a hierarchical design framework for biological systems, accelerating construction of biological systems for testing, and aiding in development of ultra-high-throughput, selection-based assays for cell factories and cell factory components.

References

- [1] Bray, D., “Protein molecules as computational elements in living cells,” *Nature*, Vol. 376, No. 6538, 1995, pp. 307–312.
- [2] Hartwell, L. H., et al., “From molecular to modular cell biology,” *Nature*, Vol. 402, No. 6761 Suppl., 1999, pp. C47–C52.
- [3] Alon, U., “Biological networks: the tinkerer as an engineer,” *Science*, Vol. 301, No. 5641, 2003, pp. 1866–1867.
- [4] Jeong, H., et al., “The large-scale organization of metabolic networks,” *Nature*, Vol. 407, No. 6804, 2000, pp. 651–654.
- [5] Shen-Orr, S. S., et al., “Network motifs in the transcriptional regulation network of *Escherichia coli*,” *Nature Genet.*, Vol. 31, No. 1, 2002, pp. 64–68.
- [6] Spirin, V., and L. A. Mirny, “Protein complexes and functional modules in molecular networks,” *Proc. Natl. Acad. Sci. USA*, Vol. 100, No. 21, 2003, pp. 12123–12128.
- [7] Savageau, M. A., “Design principles for elementary gene circuits: elements, methods, and examples,” *Chaos*, Vol. 11, 2001, pp. 142–159.

- [8] Becskei, A., and L. Serrano, "Engineering stability in gene networks by autoregulation," *Nature*, Vol. 405, No. 6786, 2000, pp. 590–593.
- [9] Becskei, A., et al., "Positive feedback in eukaryotic gene networks: cell differentiation by graded to binary response conversion," *EMBO J*, Vol. 20, No. 10, 2001, pp. 2528–2535.
- [10] Isaacs, F. J., et al., "Prediction and measurement of an autoregulatory genetic module," *Proc. Natl. Acad. Sci. USA*, Vol. 100, No. 13, 2003, pp. 7714–7719.
- [11] Mangan, S., and U. Alon, "Structure and function of the feed-forward loop network motif," *Proc. Natl. Acad. Sci. USA*, Vol. 100, No. 21, 2003, pp. 11980–11985.
- [12] Brent, R., and M. Ptashne, "A bacterial repressor protein or a yeast transcriptional terminator can block upstream activation of a yeast gene," *Nature*, 312, 1984, pp. 612–615.
- [13] Brent, R., and M. Ptashne, "A eukaryotic transcriptional activator bearing the DNA specificity of a prokaryotic repressor," *Cell*, 43, No. 3, Pt. 2, 1985, pp. 729–736.
- [14] Frankel, A. D., and P. S. Kim, "Modular structure of transcription factors: implications for gene regulation," *Cell*, Vol. 65, No. 5, 1991, pp. 717–719.
- [15] Kaern, M., et al., "The engineering of gene regulatory networks," *Annu. Rev. Biomed. Engin.*, Vol. 5, 2003, pp. 179–206.
- [16] Patten, P. A., et al., "Applications of DNA shuffling to pharmaceuticals and vaccines," *Curr. Opin. Biotechnol.*, Vol. 8, No. 6, 1997, pp. 724–733.
- [17] Kolkman, J. A., and W. P. Stemmer, "Directed evolution of proteins by exon shuffling," *Nat. Biotechnol.*, Vol. 19, No. 5, 2001, pp. 423–428.
- [18] Umeno, D., et al., "Diversifying carotenoid biosynthetic pathways by directed evolution," *Microbiol. Mol. Biol. Rev.*, Vol. 69, No. 1, 2005, pp. 51–78.
- [19] Ouzounis, C. A., and P. D. Karp, "Global properties of the metabolic map of *Escherichia coli*," *Genome Res.*, Vol. 10, 2000, pp. 568–576.
- [20] Sambrook, J., and D. W. Russell, *Molecular Cloning*, CSHL Press, 2001.
- [21] Schell, M. A., "Cloning and expression in *Escherichia coli* of the naphthalene degradation genes from plasmid NAH7," *J. Bacteriol.*, Vol. 153, No. 2, 1983, pp. 822–829.
- [22] Ensley, B. D., et al., "Expression of naphthalene oxidation genes in *Escherichia coli* results in the biosynthesis of indigo," *Science*, Vol. 222, No. 4620, 1983, pp. 167–169.
- [23] Ingram, L. O., et al., "Genetic engineering of ethanol production in *Escherichia coli*," *Appl. Environ. Microbiol.*, Vol. 53, No. 10, 1987, pp. 2420–2425.
- [24] Peoples, O. P., and A. J. Sinskey, "Poly-beta-hydroxybutyrate (PHB) Biosynthesis in *Alcaligenes eutrophus* H16," *J. Biol. Chem.*, Vol. 264, No. 26, 1989, pp. 15298–15303.
- [25] Bailey, J. E., "Toward a Science of Metabolic Engineering," *Science*, Vol. 252, No. 5013, 1991, pp. 1668–1675.
- [26] Cameron, D. C., and I. T. Tong, "Cellular and metabolic engineering—an overview," *Appl. Biochem. Biotechnol.*, Vol. 38, No. 1-2, 1993, pp. 105–140.
- [27] Stephanopoulos, G., and A. J. Sinskey, "Metabolic engineering—methodologies and future prospects," *Trends Biotechnol.*, Vol. 11, No. 9, 1993, pp. 392–396.
- [28] Nielsen, J., "Metabolic engineering: techniques for analysis of targets for genetic manipulations," *Biotechnol. Bioeng.*, Vol. 58, No. 2-3, 1997, pp. 125–132.
- [29] Bailey, J. E., "Lessons from metabolic engineering for functional genomics and drug discovery," *Nature Biotechnol.*, Vol. 17, 1999, pp. 616–618.
- [30] Stephanopoulos, G. N., et al., *Metabolic Engineering: Principles and Methodologies*, London: Academic Press, 1999.
- [31] Nielsen, J., "Metabolic engineering," *Appl. Microbiol. Biotechnol.*, Vol. 55, 2001, pp. 263–283.
- [32] Rab, R. M., et al., "Metabolic engineering," *Adv. Biochem. Eng./Biotechnol.*, Vol. 100, 2005, pp. 1–17.
- [33] Seressiotis, A., and J. E. Bailey, "MPS: an artificially intelligent software system for the analysis and synthesis of metabolic pathways," *Biotechnol. Bioeng.*, Vol. 31, 1988, pp. 587–602.

- [34] Mavrovouniotis, M. L., et al., "Computer-aided synthesis of biochemical pathways," *Biotechnol. Bioeng.*, Vol. 36, 1990, pp. 1119–1132.
- [35] Hatzimanikatis, V., et al., "Exploring the diversity of complex metabolic networks," *Bioinformatics*, Vol. 21, No. 8, 2005, pp. 1603–1609.
- [36] Niederberger, P., et al., "A strategy for increasing an in vivo flux by genetic manipulation," *Biochem. J.*, Vol. 287, 1992, pp. 473–479.
- [37] Datta, P., "Regulation of branched biosynthetic pathways in bacteria," *Science*, 165, No. 3893, 1969, pp. 556–562.
- [38] Stephanopoulos, G., and J. J. Vallino, "Network rigidity and metabolic engineering in metabolite overproduction," *Science*, Vol. 252, No. 5013, 1991, pp. 1675–1681.
- [39] Kacser, H., and L. Acerenza, "A universal method for achieving increases in metabolite production," *Eur. J. Biochem.*, Vol. 216, 1993, pp. 361–367.
- [40] Varma, A., and B. O. Palsson, "Metabolic flux balancing: basic concepts, scientific and practical use," *Bio/Technology*, Vol. 12, 1994, pp. 994–998.
- [41] Schilling, C. H., and B. O. Palsson, "The underlying pathway structure of biochemical reaction networks," *Proc. Natl. Acad. Sci. USA*, Vol. 95, 1998, pp. 4193–4198.
- [42] Segre, D., et al., "Analysis of optimality in natural and perturbed metabolic networks," *Proc. Natl. Acad. Sci. USA*, Vol. 99, No. 23, 2002, pp. 15112–15117.
- [43] Stephanopoulos, G., "Metabolic fluxes and metabolic engineering," *Metabolic Eng.*, Vol. 1, 1999, pp. 1–11.
- [44] Zupke, C., and G. Stephanopoulos, "Modeling of isotope distributions and intracellular fluxes in metabolic networks using atom mapping matrices," *Biotechnol. Bioeng.*, Vol. 10, 1994, pp. 489–498.
- [45] Lutz, R., and H. Bujard, "Independent and tight regulation of transcriptional units in *Escherichia coli* via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements," *Nucleic Acids Res.*, Vol. 25, No. 6, 1997, pp. 1203–1210.
- [46] Jacob, F., and J. Monod, "Genetic regulatory mechanism in synthesis of proteins," *J. Mol. Biol.*, Vol. 3, 1961, pp. 318–356.
- [47] Schleif, R., "Induction of the L-arabinose operon," *J. Mol. Biol.*, Vol. 46, No. 1, 1969, pp. 197–199.
- [48] Bertrand, K. P., et al., "Overlapping divergent promoters control expression of Tn10 tetracycline resistance," *Gene*, 23, No. 2, 1983, pp. 149–156.
- [49] Hillen, W., et al., "Control of expression of the Tn10-encoded tetracycline resistance genes. Equilibrium and kinetic investigation of the regulatory reactions," *J. Mol. Biol.*, Vol. 169, No. 3, 1983, pp. 707–721.
- [50] Alper, H., et al., "Tuning genetic control through promoter engineering," *Proc. Natl. Acad. Sci. USA*, Vol. 102, No. 36, 2005, pp. 12678–12683.
- [51] Kim, S. W., and J. D. Keasling, "Metabolic engineering of the nonmevalonate isopentenyl diphosphate synthesis pathway in *Escherichia coli* enhances lycopene production," *Biotechnol. Bioeng.*, Vol. 72, No. 4, 2001, pp. 408–415.
- [52] Alper, H., et al., "Identifying gene targets for the metabolic engineering of lycopene biosynthesis in *Escherichia coli*," *Metabolic Engineering*, Vol. 7, No. 3, 2005, pp. 155–164.
- [53] Looger, L. L., et al., "Computational design of receptor and sensor proteins with novel functions," *Nature*, Vol. 423, No. 6936, 2003, pp. 185–190.
- [54] Uil, T. G., et al., "Therapeutic modulation of endogenous gene function by agents with designed DNA-sequence specificities," *Nucleic Acids Res.*, Vol. 31, 2003, pp. 6064–6078.
- [55] Carrier, T. A., and J. D. Keasling, "Controlling messenger RNA stability in bacteria: strategies for engineering gene expression," *Biotechnol. Progr.*, Vol. 13, No. 6, 1997, pp. 699–708.
- [56] Wood, T. K., and S. W. Peretti, "Depression of protein synthetic capacity due to cloned-gene expression in *E. coli*," *Biotechnol. Bioeng.*, Vol. 36, 1990, pp. 865–878.

- [57] Carrier, T. A., and J. D. Keasling, "Library of synthetic 5' secondary structures to manipulate mRNA stability in *Escherichia coli*" *Biotechnol. Progr.*, 15, No. 1, 1999, pp. 58–64.
- [58] Smolke, C. D., et al., "Coordinated, differential expression of two genes through directed mRNA cleavage and stabilization by secondary structures," *Appl. Environ. Microbiol.*, Vol. 66, No. 12, 2000, pp. 5399–5405.
- [59] Smolke, C. D., et al., "Controlling the metabolic flux through the carotenoid pathway using directed mRNA processing and stabilization," *Metab. Eng.*, Vol. 3, No. 4, 2001, pp. 313–321.
- [60] Winkler, W. C., et al., "Control of gene expression by a natural metabolite-responsive ribozyme," *Nature*, Vol. 428, No. 6980, 2004, pp. 281–286.
- [61] Tucker, B. J., and R. R. Breaker, "Riboswitches as versatile gene control elements," *Curr. Opin. Struct. Biol.*, Vol. 15, No. 3, 2005, pp. 342–348.
- [62] Martin, V. J., et al., "Engineering a mevalonate pathway in *Escherichia coli* for production of terpenoids," *Nat. Biotechnol.*, Vol. 21, No. 7, 2003, pp. 796–802.
- [63] Hale, R. S., and G. Thompson, "Codon optimization of the gene encoding a domain from human type 1 neurofibromin protein results in a threefold improvement in expression level in *Escherichia coli*," *Protein Exp. Purif.*, 12, No. 2, 1998, pp. 185–188.
- [64] Tong, I.-T., et al., "1,3-Propanediol production by *Escherichia coli* expressing genes from the *Klebsiella pneumoniae* dha Regulon," *Appl. Environ. Microbiol.*, Vol. 57, No. 12, 1991, pp. 3541–3546.
- [65] Laffend, L. A., et al., "Bioconversion of a fermentable carbon source to 1,3-propanediol by a single microorganism," 1997.
- [66] Emptage, M., et al., "Process for the biological production of 1,3-propanediol with high titer," 2003.
- [67] Carr, P. A., et al., "Protein-mediated error correction for de novo DNA synthesis," *Nucleic Acids Res.*, Vol. 32, No. 20, 2004, p. e162.
- [68] Kodumal, S. J., et al., "Total synthesis of long DNA sequences: Synthesis of a contiguous 32-kb polyketide synthase gene cluster," *Proc. Natl. Acad. Sci. USA*, Vol. 101, No. 44, 2004, pp. 15573–15578.
- [69] Tian, J., et al., "Accurate multiplex gene synthesis from programmable DNA microchips," *Nature*, Vol. 432, 2004, pp. 1050–1054.
- [70] Itaya, M., et al., "Combining two genomes in one cell: Stable cloning of the *Synechocystis* PCC6803 genome in the *Bacillus subtilis* 168 genome," *Proc. Natl. Acad. Sci. USA*, 102, No. 44, 2005, pp. 15971–15976.
- [71] Menzella, H. G., et al., "Combinatorial polyketide biosynthesis by de novo design and rearrangement of modular polyketide synthase genes," *Nature Biotechnol.*, Vol. 23, No. 9, 2005, pp. 1171–1176.
- [72] Liao, J. C., "Custom design of metabolism," *Nature Biotechnol.*, Vol. 22, No. 7, 2004, pp. 823–824.

PART VI

Integration: Applying Biology's Designs and Principles in Engineering

The Three Faces of DNA/RNA Sequence Hybridization

Olgica Milenkovic

12.1 Introduction

Sequence hybridization and self-hybridization are two important biological processes that involve macromolecules responsible for storing and transmitting genetic information. The basic principle underlying hybridization is the chemical affinity of bases in single DNA and RNA strands to form hydrogen bonds with their complementary bases, defined in terms of the Watson-Crick rule. By forming such bonds, paired bases generate planar or spatial structures that are comprised of two complementary strands or one single DNA or RNA strand. These structures increase the overall stability of the molecules and also play an important role in regulating various cellular functions, including DNA editing and post-transcriptional gene silencing. In certain cases, specific self-hybridization patterns in DNA sequences can represent precursors to sequence breakage and are closely associated with the onset of genetic diseases such as cancer.

Due to its simple governing properties, sequence hybridization is becoming an increasingly important technique used in a new generation of parallel computing, storage, and data processing nanodevices. Hybridization is the basic reaction supporting the operation of modern DNA-based computers, DNA logical circuits, and autonomous automata capable of controlling the cell cycle and regulating gene expression levels. It is also the operational principle of DNA microarrays, or genetic chips. Genetic chips are one of the most widely employed diagnostic tools, used to generate extensive data for comparative cell studies. Self-hybridized DNA and RNA structures have also found applications in processes as diverse as nanoparticle assembly and DNA data encryption.

Besides its chemical and physical properties, sequence hybridization has distinctly combinatorial features that can be exploited to improve the performance of systems using hybridization as its core functionality. These combinatorial features represent the starting point for the design of DNA codes and for the enumeration of DNA and RNA planar and tertiary structures. DNA coding techniques increase both the efficiency and reliability of DNA/RNA systems employing sequence hybridization, which makes them a valuable tool for developing robust DNA

computers and microarrays. Similarly, enumeration methods play an important role in designing statistical experiments involving DNA/RNA sequence self-hybridization.

It is the goal of this chapter to introduce the reader to three important aspects of the DNA and RNA sequence hybridization phenomena. The first aspect is concerned with the versatility of hybridization techniques occurring within a living cell, and with the various regulatory mechanisms controlled by sequence hybridization and self-hybridization. Hybridization experiments can also be performed in controlled in vitro environments, which allows for developing man-made systems capable of mimicking some of the functions performed by living cells. The second aspect of sequence hybridization is concerned with technological applications of this biological process, including DNA computers, DNA microarrays, and DNA self-assembly. The third aspect is concerned with problems related to the issue of efficiently controlling the accuracy of sequence hybridization, which can be addressed in terms of invoking results from combinatorics, and the theory of error-control and constrained coding.

The chapter is organized as follows. In Section 12.2 a brief introduction to hybridization and self-hybridization is provided. Section 12.3 details a selected set of biological processes in which hybridization plays an important role. Section 12.4 is devoted to the description of several emerging biotechnical systems utilizing sequence hybridization as their main operational principle. In Section 12.5, hybridization and self-hybridization problems are cast in a coding-theoretic framework. Problems such as the design of codes for DNA computers and RNA/DNA motif enumeration are addressed in terms of ideas borrowed from channel and constrained coding theory. Concluding remarks are presented in Section 12.6.

12.2 A Short Introduction to DNA/RNA Sequence Hybridization and Self-Hybridization

This section contains some basic combinatorial definitions and concepts relating to hybridization and self-hybridization (folding) of DNA and RNA sequences. For more details regarding the biochemical nature of these processes, the interested reader is referred to [1].

DNA and RNA molecules consist of three types of building blocks: sugar molecules, phosphate groups, and bases. The sugar and phosphate groups are strung together in an alternating fashion, forming the so-called *sugar-phosphate* backbone of the molecules. The phosphate groups are the same in DNA and RNA strands, but the five-carbon sugars differ: in the first case, the sugar is deoxyribose, while in the second case it is ribose. DNA and RNA strands are assumed to have an orientation due to the asymmetric structure of their sugar-phosphate backbones. One end of the strand is usually designated as the 3' end (referring to the index of the carbon molecule to which the terminal phosphate group is attached), while the other is similarly referred to as the 5' end. An illustration of this structure and the described terminology is given in Figure 12.1. The bases in DNA strands can be partitioned into two groups of elements, known as *purines* and *pyrimidines*. Purines include the bases adenine (A) and guanine (G), while pyrimidines include the bases thymine (T)

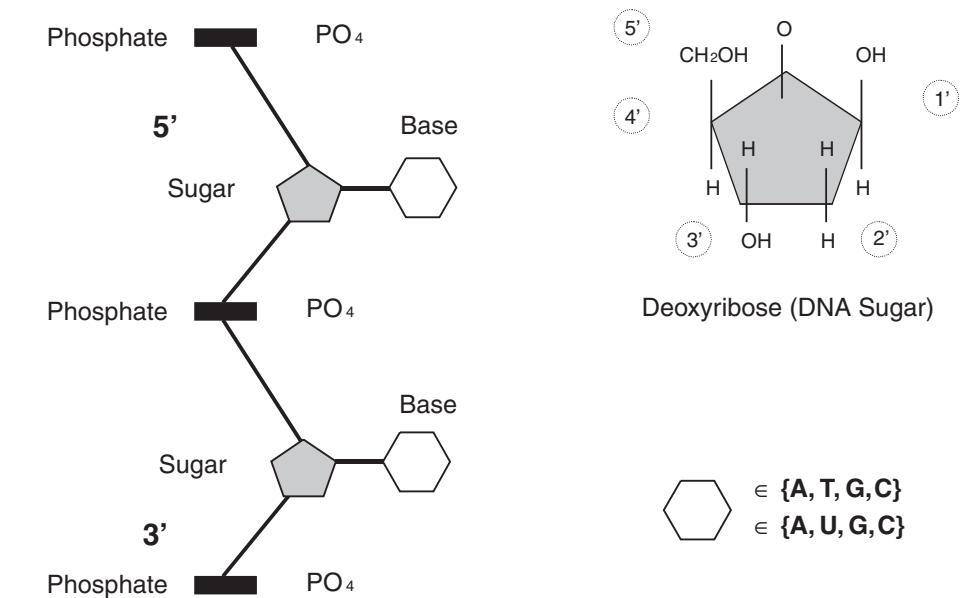


Figure 12.1 DNA/RNA sequence structure.

and cytosine (C). Similarly, bases in RNA molecules are of the same type, with the exception of the base T being replaced by uracil (U).

Since the sugar-phosphate backbone of DNA and RNA molecules has a fixed structure, at the first level of abstraction, DNA and RNA strands can be represented by oriented words over the four-letter alphabet of their bases. At the second level of abstraction, DNA and RNA molecules can be viewed as geometrical structures—more specifically, as two-dimensional or three-dimensional shapes. Such shapes arise from the affinity of the bases to bond with each other and form stable folded configurations. Most frequently, a simple bonding rule is observed: G binds to C in terms of three hydrogen bonds, and vice versa, while A binds to T (or to U, in the case of RNA strands) in terms of two hydrogen bonds, and vice versa. These bonding rules are known as *Watson-Crick (WC) complementation*.¹ More formally, if $\bar{X}$ denotes the WC complement of X , then $\bar{\bar{A}} = \text{T}$, $\bar{\bar{T}} = \text{A}$, $\bar{\bar{G}} = \text{C}$, $\bar{\bar{C}} = \text{G}$.

If base-pairings occur between two individual strands with opposite direction, the resulting process is referred to as *hybridization*. The genome itself is organized in terms of two DNA strands hybridized so as to form a double helix that is coiled in the cell's nucleus of eukaryotic species. If the base-pairing occurs among bases within the same strand, the process is termed *self-hybridization* or *folding*. Self-hybridization converts a one-dimensional strand into a two- or three-dimensional strand. The formations obtained in this way are usually referred to as the *secondary* and *tertiary* structure of the sequence, respectively.² The *primary* structure of a DNA/RNA strand is its one-dimensional sequence of bases. Hybridization can be complete or incomplete: in the latter case, only subsets of the bases on the strand

1. Other semistable forms of base-binding can be found in genetic sequences as well, but due to their significantly smaller incidence rates, they are not considered in this chapter.
2. Henceforth, the terms *folding* and *self-hybridization* are both used to describe the formation of secondary structures of DNA/RNA strands.

bind with each other. Folding is usually achieved in terms of incomplete binding between bases on the same strand. An example of two imperfectly hybridized DNA strands and an example of a two-dimensional RNA folding pattern are shown in Figure 12.2(a) and (b) respectively.

The RNA sequence shown in Figure 12.2(b),

3'-AACCCGCCCCUUGGGGGGACAUUCUAAGGUCGAGG-5'

folds in two dimensions, and its secondary structure involves one *dangling strand* (consecutive unpaired bases at either the 3' or 5' end of the sugar-phosphate backbone), five *stem (helical) regions* including perfectly matched Watson-Crick complementary subsequences, and five *loops*.

Loops can be broadly classified into four classes: *hairpins*, *internal*, *branching*, and *bulge* loops. A hairpin loop is an exterior loop connected to one stem region only, an internal loop connects two stems, and a branching loop has connections to at least three stems. A bulge is a protrusion between two stems that appears only on one side of the folded structure. Henceforth, we refer to all the aforementioned structures simply as loops.

Among the most frequently encountered single-stranded DNA and RNA secondary structures in a cell are *hairpins* and *cruciforms*, shown in Figure 12.3(a) and (b) respectively.

A folded shape is considered a tertiary structure if, in addition to stems and loops, it includes binding patterns known as *pseudoknots*, *loop-to-loop interactions*, and *stem interactions*.

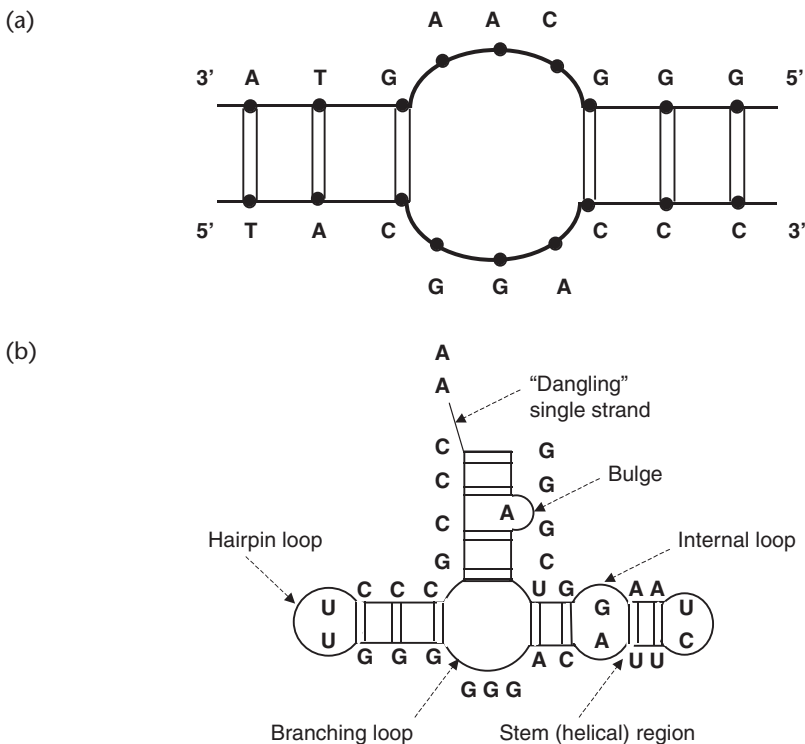


Figure 12.2 (a) Two imperfectly hybridized DNA strands; (b) RNA secondary structure.

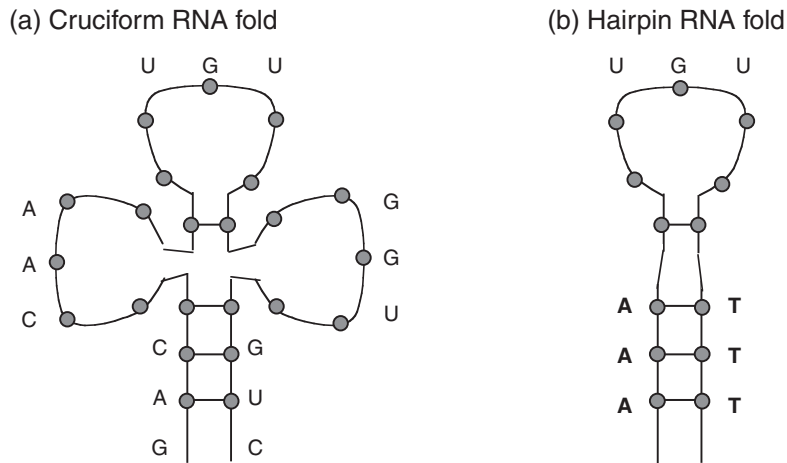


Figure 12.3 Two RNA secondary structures.

Very short single-stranded DNA and RNA sequences usually do not fold, which makes them amenable for use in systems for which secondary and tertiary structure formations are undesired. Short single-stranded sequences are called *oligonucleotide sequences*, or *oligosequences* (oligostrands).

12.3 DNA/RNA Sequence Hybridization: A Biological Point of View

While DNA sequences have the primary function of storing the instructions for building and maintaining a cell, RNA strands are involved both in the process of storing genetic material and in directing the cell's life cycle by performing a multitude of regulatory functions [1]. RNA sequences serve as carriers of information stored in the DNA code (messenger RNA, or mRNA), transporters of protein building blocks (transfer RNA, or tRNA), structural units that facilitate the process of translation (ribosomal RNA, or rRNA), or general regulatory and functional entities (functional RNA, or fRNA). In all these settings, RNA strands tend to fold. The mRNA molecules usually fold to achieve higher structural stability and to allow or prohibit access of certain enzymes to their bases, while tRNAs fold in a characteristic three-arm shape to which amino acids and anticodons, recognizing base triples on mRNA molecules, are attached. The rRNAs represent a special class of ribozymes responsible for mediating the process of protein synthesis through controlled RNA-RNA and RNA-protein interactions. Both ribozymes and regulatory RNA strands fold in order to bind to other molecules. The exact nature of the control mechanisms involving self-hybridized functional RNA and DNA molecules is described in more detail below.

12.3.1 Functional RNA Molecules

The *Central Dogma* of genetics asserts that the information contained in DNA regions known as genes can be unidirectionally transferred to mRNA strands in a

process called *transcription*. Genes consist of subsequences known as *exons* and *introns*. In the first step of transcription, a complete coding region is transcribed into an RNA strand called pre-mRNA. Pre-mRNA is converted into mature mRNA through the process of *splicing*, which consists of removing substrands that correspond to introns of genes (note that the boundaries of introns are not strictly defined so that splicing can have several different forms). A gene is said to be *expressed* if it is transcribed into mature mRNA, and its *expression level* corresponds to the speed of the underlying transcription process or, alternatively, to the concentration level of its mRNA. Upon completion of transcription, mature mRNA molecules are used to guide protein synthesis through the ribosome complex that includes rRNA molecules, in a process known as *translation*. During translation, every triple of bases in mRNA strands is translated into a specific amino acid, and a sequence of such amino acids represents a polypeptide chain that is an integral part of the protein structure.

The role of RNA is not confined only to mediating the process of protein synthesis: RNA sequences can also perform many functions in a cell, and in this case they are termed functional RNAs, or fRNAs [2–4]. The discovery of fRNA purports the theory that RNA molecules were both the initial storage media for genetic material as well as the first forms of life, only to be replaced in later stages of evolution by DNA and proteins.

There exist many classes of fRNAs, the most interesting examples of which include *ribozymes*—catalytic RNA sequences that can assume at least eight different forms—as well as *riboswitches* and *microRNA* (miRNA) that serve as self-regulatory or regulatory elements during translation respectively.

Ribozymes (catalytic RNAs) are rare molecules first described in [2], known to catalyze important chemical reactions such as RNA cleavage, RNA synthesis, to support specialized activities of the ribosome, or to act as chaperon enzymes involved in protein folding.

Riboswitches are short fRNAs embedded within mRNA strands that are able to regulate the function of their own host mRNA [3]. Regulation is achieved in terms of enabling or disabling the binding of target molecules to the mRNA strand. As a consequence, one of several possible actions can be performed under the guidance of riboswitches: self-cleavage (or other structure-alternating changes), premature termination of transcription, and initiation of translation.

MicroRNAs are short RNA strands, well preserved in many species, known to play an important role in regulating the expression levels of genes [4]. MicroRNAs are also implicated in processes underlying parts of the cell cycle, cell differentiation, and in particular, tumorigenesis.

DNA/RNA aptamers are oligostrands with secondary and tertiary structures that allow them to bind to specific organic molecules, such as proteins [5]. Aptamers are often used as “therapeutic antibodies,” due to the fact that they can be designed to bind to various enzymes and viral subunits (for more technical details regarding aptamers, the interested reader is referred to [6]). Aptamers are usually identified in terms of randomly generating large numbers of RNA sequences, mixing them with a pool of targeted protein structures, and “filtering out” bonded RNA-protein structures. This approach is known under the name *SELEX* (Systematic Evolution of Ligands by Exponential Enrichment) [7].

12.3.2 Gene Silencing and RNA Interference

As already described, fRNA strands are known to hybridize or fold into planar and tertiary structures in order to increase their stability, and once bonded, the bases in RNA strands tend to reduce their chemical activity or become inaccessible for binding enzymatic complexes. It was recently discovered that it is possible to reduce or completely silence the activity of mRNA molecules by using complementary hybridization and hybridization-guided cleavage [8–10]. More specifically, in one possible scenario, a short RNA sequence is hybridized to its complementary mRNA strand, thereby blocking the process of translation. In a second scenario, a subsequence of the mRNA strand is recognized through hybridization with a complementary RNA molecule that is either already present in the cell or artificially introduced into it. The latter RNA strand is part of a complex that, once exposed to the mRNA, tends to cleave it in the position indicated by the hybridized duplex. Processes of this form belong to the class of post-transcriptional gene silencing (PTGS) mechanisms that result in rapid degradation of the partly hybridized mRNA strands. PTGS can also be initiated by introducing short, double-stranded RNA (dsRNA) sequences into the cell, known as *small interfering RNA* (siRNA). PTGS involving siRNA is known as RNA interference (RNAi) [8], and siRNAs represent parts of the so-called RISC (RNA-induced silencing complex). The RISC also includes proteins, and is capable of recognizing and disabling endogenous mRNAs with subsequences complementary to the siRNA. It is interesting to point out that RNAi is assumed to be closely tied to the process of inducing *nonsense-mediated decay* [8], governed by a proofreading mechanism that protects the cell from translating erroneously transcribed mRNAs.

12.3.3 RNA Editing and Re-encoding

RNA editing is a biomolecular process involving various forms of mRNA, rRNA, tRNA, and other regulatory RNA molecules, and resulting in a modification of the original primary and secondary structure of the strands [11]. Editing is a mechanism for altering the content of a sequence in terms of base substitution, insertion, and/or deletion. Editing provides for increased diversity of protein structures, along with the previously described process of alternate splicing (i.e., alternate interpretation of intron and exon boundaries in a DNA strand), and the process of *5' capping* (i.e., insertion of a modified G base into the 5' end of pre-mRNA). For coding RNA, editing usually occurs after transcription and before translation, and in some cases the presence of introns in pre-mRNA is crucial for proper editing. More precisely, editing sites are most frequently found in well-preserved intronic regions, which upon editing may become parts of the coded strand [12]. Introns in pre-mRNA lead to special folding structures of the molecules that are recognized by the editing enzymes: this phenomena is especially pronounced in strands that contain subsequences known as *Alu repeats* (near-perfect repeats of one or more short DNA sequences) and *inverted repeats* (two different subsequences on the same strand that are Watson-Crick complements of each other). Alternative forms of editing, involving DNA rather than RNA strands, have been documented in [13], primarily in connection with DNA damage repair caused by erroneous DNA proofreading

mechanisms. In this respect, RNA editing is an important part of the error-correcting process for genetic sequences, the malfunctioning of which can lead to various forms of cancer and neurological disorders.

As already pointed out, editing consists of base substitutions, insertions, and deletions, or a combination thereof [11]. The best-documented example of RNA substitution editing is the editing process of the mRNA strand of the so-called *human APOB gene*. This gene is expressed both in the intestines and the liver, but in the former organ the molecule is edited in the next to last codon (triple). Through the help of an enzyme, the base C is converted into the base U. In the case where this operation is performed on the triple CAA (encoding for the amino acid glutamine), the resulting encoding becomes the STOP codon—a signal for terminating the process of translation. This type of editing leads to different functional properties of the corresponding proteins: in the liver, the unedited mRNA sequence produces a protein aiding the transport of lipids in the bloodstream, while in the intestines the edited mRNA introduces a protein helpful for absorption of lipids. There are also eight known mammalian genes involved in so-called *A-I mRNA editing*, during which the base A is transformed into ionine (I), which is subsequently interpreted as the base G. The A-I editing enzyme recognizes a special RNA hairpin folded structure, which signals that editing has to be performed at the given location.

Of special importance is the process of insertion and deletion editing, since it provides for more editing diversity in terms of frame shifts. Insertion and deletion editing involves two classes of RNA molecules: mRNA and guide RNA (gRNA) structures. gRNA, as its name suggests, is responsible for guiding the editing process, and it serves as a type of a “biological mask” describing which bases are to be inserted into or deleted from the underlying mRNA molecule [13].

Consider the example below, where both the mRNA and gRNA strands are stretched in order to illustrate the principles of gRNA editing:

gRNA: 3' – AUUUUGCCCAA **GUA** CCCU **UU** AAGGGCCCCC – 5'
mRNA: 5' – GGUU GGGA TTCCC – 3'

At the starting and ending point of the editing region (denoted by boldface base symbols), perfect hybridization patterns involving the sequences **GGUU** and **TTCCC** are encountered. The gRNA strand hybridizes with its target mRNA in an imperfect manner, insofar that only a subset of the bases in the gRNA bonds with complementary bases in the mRNA molecule. In the example, this leads to the creation of two bulges, involving the sequences *GUA* and *UU* in the gRNA molecule. The backbone of the mRNA molecule is consequently cut in order to allow for extending the sequence in terms of incorporating the reverse complements of the bulge subsequences in the gRNA strand, as shown below:

gRNA: 3' – AUUUUGCCCAA **GUA** CCCU **UU** AAGGGCCCCC – 5'
mRNA: 5' – GGUU|CAU|GGGA|AA|TTCCC – 3'

Hence, the mRNA strand is edited by insertion. In some other plausible editing scenarios, bulges may arise within the mRNA strand. In this case, the mRNA strand

undergoes a deletion editing process, resulting in the removal of the bases in the bulges.

RNA editing is a means of achieving genetic diversity: a few gRNA can be used to edit and modify a large number of mRNA molecules, resulting in a number of functionally related protein structures. In this aspect, RNA editing has a very similar function as the process of alternative splicing [1].

12.3.4 Fragile DNA Regions and Secondary Structures

The double helix, formed by two Watson-Crick (WC) complementary single-stranded DNA units, represents a special organizational structure of nucleotides that allows for parallel replication of the strands and for error correction of mutated bases on one strand based on the information provided by the other strand [1]. The double-helix formation is only one possible hybridized form for DNA strands: there is increasing evidence that the two constituent DNA stands tend to form folded patterns that protrude from the helix [14, 15]. These patterns are recognized to have several functions, one of them being to act as “signals” for certain enzymes operating on DNA strands to attach to the protruded unit and its adjacent double-helix DNA formation. Usually, these protrusions are of the form of simple hairpins or cruciforms [14], as shown in Figure 12.3.

A more ominous role of folded protrusions is the one they play in the development and progression of illnesses, such as the hereditary *Huntington*, *Fragile X*, and *Friedreich’s ataxia* disease, the so-called Emanuel syndrome, as well as certain forms of cancer [16–20]. In the first case, it is known that *microsatellite DNA*—repetitive subsequence units in DNA strands present in most eukaryotic organisms (including Alu repeats)—can cause double-helix instabilities that lead to the formation of protrusions. Furthermore, microsatellite DNA units have a tendency to change their number during the replication process, either due to replication slippage or errors in DNA proofreading and mismatch repair mechanisms, or a combination of these two and some other error mechanisms [1, 17]. While repeats of patterns involving two nucleotides are associated with certain forms of cancer, hereditary diseases like Huntington’s disease are associated with repeats involving three bases. Three triples are identified to have such effects, namely CAG, CGG, and GAA. Although for most species the different mechanisms behind the change in the microsatellite repetition length are still not well understood, it is widely believed that secondary structures, such as hairpins, formed by the triples of the form described above, contribute to the strands’ fragility and faulty DNA replication. These results are supported by in vitro and in vivo studies described in [16–19]. Furthermore, the secondary structures formed by the triplet repeats are known to prevent normal progression of DNA replication itself.

Emanuel’s syndrome is caused by an exchange of genetic material (i.e., by *translocation*) involving chromosome 11 and the small, compositionally unstable chromosome 22 [20]. While these translocations can be normal in an individual, they can lead to the appearance of an extra “mixed” chromosome 11 and 12 in its offspring. In some respect, this is similar to the Down syndrome phenotype [1], caused by an extra copy of a chromosome, in this case the 21st chromosome. Individuals with this syndrome suffer from severe mental retardation and heart defects.

The translocation present in Emanuel's syndrome patients is achieved by a mechanism involving breakage of both chromosome 11 and chromosome 22 along fragile DNA regions consisting of *palindromic repeats*. Palindromic repeats are DNA subsequences that have the property that, when read from left-to-right produce the WC complement of the subsequence when read from right-to-left. The palindromes that represent the "fault lines" for breakage have a high concentration of the bases A and T. As with many other disease scenarios, formations of hairpins and cruciforms in these regions are the actual cause of double-stranded breakage. Palindromes are also known to occur in mitochondrial DNA involving short, repetitive G- and C-rich regions that can fold into secondary structures involving two stem-hairpin formations [15]. Similar mechanisms exist in various types of cancer, where substantial chromosome mixing occurs. Fragile sites can also cause chromosome deletions associated with diseases such as Jacobsen syndrome [17]. The deletion breakpoints associated with this disease are all known to contain CGC repeats. This provides evidence for the nonrandom chromosome deletion theory, which also supports the *fragile chromosome breakage model* used for describing evolutionary changes of DNA sequences [21].

12.4 DNA/RNA Sequence Hybridization: A Technological Point of View

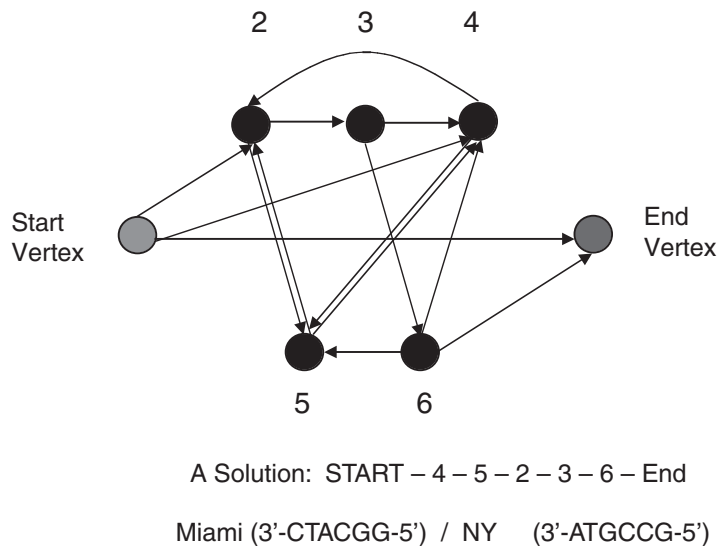
The biological processes described in Section 12.3 can be used as templates for designing man-made DNA systems capable of performing tasks ranging from parallel computing to controlling the cell cycle. This section provides an overview of such biotechnological systems, with a special focus on those that utilize genetic sequence hybridization and self-hybridization processes, and more specifically, RNA editing and control of gene expression through RNA interference.

12.4.1 DNA Computers

In 1994, Leonard Adleman [22] demonstrated that DNA sequence hybridization can be used as the design principle of a new class of powerful parallel computing devices. Adleman's experiment marked the birth of *DNA computers*, and opened the possibility of data processing at the level of biological molecules which operate as non-von Neuman, stochastic machines, to which the limitations of typically sequential silicon-based computers do not apply.

In its original setting, DNA computing was envisioned as a method for solving computational problems that are impossible to unravel using traditional computing methods. DNA computers can perform this task through massive parallel operations on the molecular nanoscale level and by using low-power molecular hardware and software systems.

Adleman considered a simple instant of the directed traveling salesman (DTS) problem on a graph with seven nodes and fourteen edges, shown in Figure 12.4 [22]. The DTS is a nonpolynomial deterministic (NP)-complete combinatorial problem concerned with finding a path through a directed graph that starts and ends in a given node and visits each node in the graph exactly once. In order to solve this problem, a set of seven oligonucleotide DNA sequences of length six was selected



Route (Edge): second half of codeword for Miami (CGG), first half of codeword for NY (ATG): 3'-CGGATG-5', WC complement of this word: 5'-GCCTAC-3'

Figure 12.4 Adleman's traveling salesman problem. (Adapted from [22].)

in an appropriate manner. These sequences were used to describe the cities that the traveling salesman was to visit. Another set of fourteen oligosequences was selected to denote the roads available to the salesman, that is, the edges of the directed graph. The DNA sequence assigned to an edge connecting two cities was chosen to represent the WC complement of the concatenation of the last three bases of the sequence of the origin city and the first three bases of the sequence assigned to the destination city (see Figure 12.4 for an illustrative example).

The computing process itself was performed by controlled hybridizations of oligonucleotide strands representing cities and roads in an alternating manner.

Following Adleman's initial work, other classes of combinatorial questions were addressed on various forms of DNA computers exploiting some type of biological property related to hybridization. Among the implementations using the same hybridization principles as described in [22] are a 20-variable *satisfiability problem* for Boolean forms on three variables (3-SAT) [23] and a special case of a *nonattacking knights* problem [24].

An alternative to DNA computing based on hybridization principles only was describe in [25], where deoxyribozymes (artificial DNA-based enzymes mimicking the function of ribozymes) were used to construct logic gates [25–27]. The functions of ribozymes were made to match basic operations of the central processing unit, such as addition, bit-shifting, as well as AND, OR, NOT, and NOR logical operations. As an illustrative example, two ribozyme structures frequently used for the construction of NOT or controlled NOT gates are shown in Figure 12.5 [25–27].

The ribozyme E6 consists of an active core and a loop. In the form shown, it can be used to denote one bit value, say "0." If the ribozyme E6 is exposed to oligonucleotide strands that are WC complementary to the sequence in its hairpin loop, it changes its shape so that the loop sequence hybridized with the oligostrand

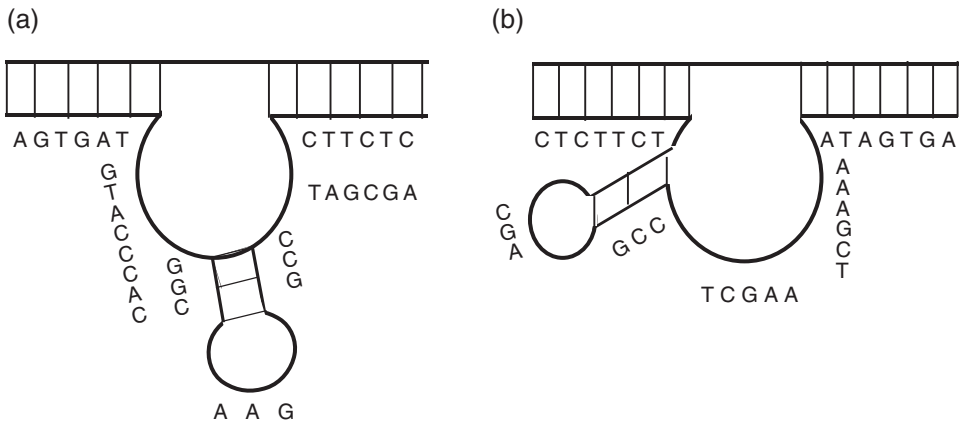


Figure 12.5 Secondary structures of the E6 and 8-17 deoxyribozyme. (Adapted from [26].)

“opens up.” The unfolded pattern of the ribozyme is used to denote a bit value “1.” Deoxyribozyme-based logic components are used as integral parts of an intelligent, interactive DNA system, called MAYA [28]. MAYA is the first known interactive DNA-based system capable of playing games such as tic-tac-toe.

An alternative form of DNA logic was developed in [29], where the implementation of a self-assembling nanoscale transistor was described. The transistor consists of a DNA strand with an appropriately attached carbon nanotube. The nanotube is equipped with nanowires to which voltage can be applied to switch the transistor on or off. For the purpose of efficient manufacturing, the combination of DNA strands and carbon nanotubes is modified by the inclusion of certain bacterial proteins that allow for fast self-assembly and bonding of the biological components of the system.

Although it is very unlikely that DNA-based computing systems will to a large extent replace electronic devices, there exist special applications for which they represent an attractive alternative or the only available option for future development. Among the most prominent such applications are DNA-based storage [30], biosensing [31], molecular signal processing and communication [32], and intracellular disease diagnostics and treatment [33]. The first prototype of a practical “smart drug” based on a DNA computer was recently implemented in vitro [33]. Smart drugs operate by detecting changes in the mRNA fingerprint of cancerous cells and by employing PTGS techniques. Here, fingerprinting refers to a set of processes aimed at detecting the presence and concentration of specific mRNA molecules in the cell. Since mRNA molecules carry the information for protein encoding, they can serve as indicators of the activity of their corresponding genes. If, for example, mRNA sequences carrying the “messages” of genes known as PPAP2B and GSTP1 are underrepresented (underexpressed), while mRNA sequences carrying the “messages” of genes known as PIM1 and HPN are overrepresented (overexpressed) within a cell in the prostate, there exists a high probability that the cell is undergoing cancerous changes. In this case, the DNA computer is instructed to administer a DNA oligosequence—GTTGGTATTGGACATG—that inhibits the generation of a protein encoded for by the MDM2 gene. This inhibition is achieved through

complementary hybridization between the DNA “drug” sequence and its WC subsequence on the target MDM2 mRNA, which either blocks the mRNA’s function or mediates its destruction [33]. Alternatively, mRNA can be deactivated by catalytic RNAs such as the hammerhead ribozyme [27]. As already pointed out, ribozymes have their own RNA cleavage system and can therefore operate without the use of any intracellular molecules. Deoxyribozymes will probably be used in future gene silencing DNA computers due to their ability to efficiently suppress the activity of any target gene. However, as with other oligonucleotide-based strategies, future improvements of this approach may depend on how efficiently one can provide for the accessibility of a target that is folded into its own secondary structure and embedded within a complex cellular environment.

DNA computers can also be designed to perform the operations of a universal Turing machine. This can be accomplished by implementing a universal DNA computer in terms of *DNA tiles*, two-dimensional DNA structures that can bond with other such structures under a certain set of rules. Usually, the tiles are represented in the form of squares, which have the property that their edges are finitely colored. The tiling rules in this case reduce to the simple requirement that two tiles can attach to each other if and only if the colors of their aligned edges are the same. For example, the first of the four tiles shown in Figure 12.6 can attach itself in the horizontal direction to the third tile, since the right side of the first and the left side of the third tile are both green (G).

Seeman and Winfree [34–36] showed that one can construct DNA tiles by using four incompletely hybridized DNA strands that form a cross structure, shown in Figure 12.6. The cross formation has four unhybridized DNA sections, termed *sticky ends*. Sticky ends are used to encode the color of the edges of DNA tiles. “Color matching” is accomplished in terms of hybridization between pairs of WC complementary sequences that describe the same color. Based on a well-known result of Wang [37], which asserts that it is possible to translate any Turing machine into a set of Wang tiles, DNA tiles can be used as building blocks of universal DNA computers.

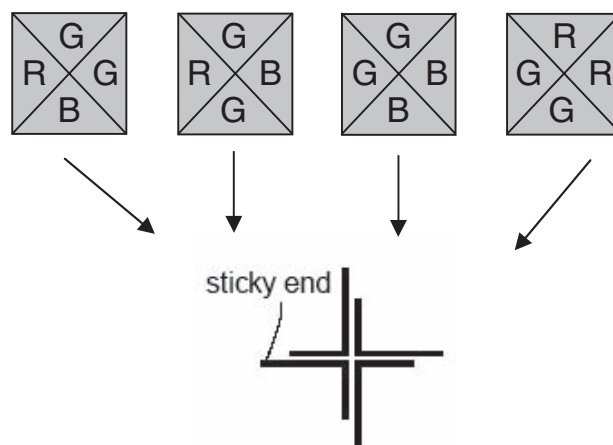


Figure 12.6 DNA Wang tiles and color coding through sticky ends.

12.4.2 DNA Microarrays

Genetic sequences, such as genes, RNA, proteins, and various regulatory elements represent parts of complicated large-scale networks. Properties of such networks can be inferred by measuring the expression levels of genes within one or multiple cells in an organism. Determining these concentrations is a nontrivial process that is largely facilitated by DNA microarrays (gene chips) [38, 39]. DNA microarrays are two-dimensional arrays of spots containing oligonucleotide DNA sequences corresponding to gene segments. The mRNA molecules extracted from a cell are converted into complementary DNA (cDNA), tagged with fluorescent markers, and then distributed on a microarray substrate. Due to the fact that DNA strands have a strong affinity to hybridize with each other, cDNA strands attach themselves to spots containing their WC complementary sequences. Which sequences succeeded in hybridizing with their complements can be detected by illuminating the chip by laser light of a given wavelength and measuring the intensity of fluorescence of the cDNA tags.

The process of DNA chip manufacturing pioneered by Affymetrix is based on photolithographic VLSIPS (Very Large Scale Immobilized Polymer Synthesis) methods, which allow for parallel production of multiple arrays on the same wafer [38]. Each chip consists of a regular grid of spots at which a large number of customer-selected DNA oligosequences (termed *probes*) are placed. These DNA strands usually represent judiciously chosen subsequences of genes. The microarray fabrication process starts with the creation of a two-dimensional wafer on which certain molecules, termed *linkers*, are placed in a regular grid of spots. Linkers are equipped with photolabile protective groups that render them inactive in the absence of external light stimuli. At each step of the production process, a *mask* is designed that specifies a subset of spots to be used during that production stage. The mask is carefully imposed over the wafer and the system is illuminated. In the presence of light, the protective groups of linkers dissolve, allowing the linkers to become operational. After this activation step, a solution containing a large concentration of the same DNA base—either A or T or G or C—is dispensed on the wafer. This allows the nucleotides to bind to active linkers, creating the first base of a selected set of DNA probes to be synthesized on the chip. Each of the added nucleotides also contains a photolabile protective group that does not allow for multiple bases to attach to the linkers or to each other. The process is repeated an appropriate number of times by choosing at each step a different mask and a different nucleotide solution to be added to the wafer.

The order of the bases used in the production process is referred to as the *base schedule*. Usually, the schedule represents an appropriate number s of periodic repetitions of the four DNA bases, denoted by ATGC^s. An example of a base schedule and masking process is shown in Figure 12.7. The masks for the five spots are represented by the columns of the array, where a black rectangle corresponds to an exposed area and a white rectangle corresponds to a masked area. The DNA sequences generated at each of the five spots in the figure are AATC, GGCG, TAAT, CTGA, and ACAA, respectively.

The design process based on the periodic schedule shown in Figure 12.7 is *synchronous*, since each spot can be exposed to light only once during a period of length four. One can also use an *asynchronous* schedule, in which one spot can be exposed to light multiple times within a period.

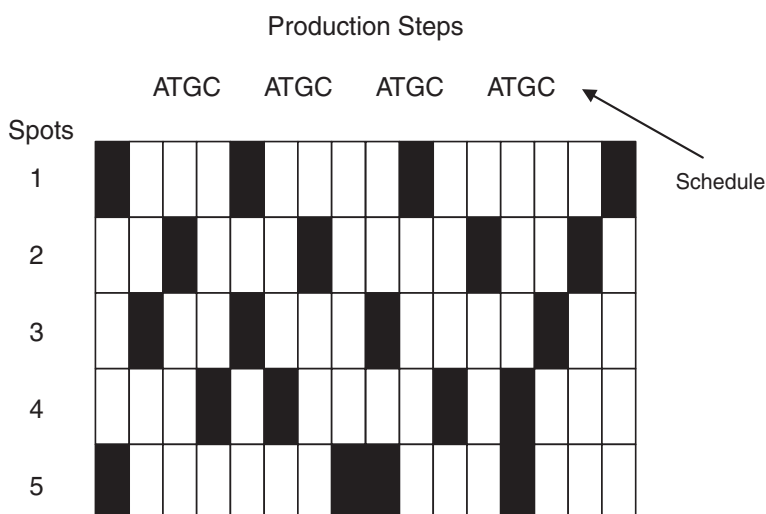


Figure 12.7 Base scheduling in the VLSIPS DNA microarray design process.

12.4.3 DNA Cryptography

Hybridization of DNA strands can be used as the governing principle behind architectures designed for the purpose of biological data hiding (steganography) and cryptography [40–43]. In such applications, hybridization allows for efficient encryption and decryption and it potentially offers security levels that may not be matched by standard cryptographical methods.

The conceptually simplest biological cryptographic methods are based on one-time pads, known to be the only secure method in classical cryptography [42]. What distinguishes DNA one-time pads from their classical counterparts is that DNA key and message sequences can be significantly longer than their classical counterparts: DNA is an extremely compact media, since its base symbols are only spaced 0.35 nm apart.

In a one-time pad system, in order to securely transmit a message of a given length, one has to select a random key (one-time pad) sequence of at least the same length. This key sequence is then either added to the message, where for each message a new key is selected randomly and independently from the previous choices for the keys, or the key is used to describe a one-to-one mapping from the set of messages to cyphers. In the case of DNA one-time pad systems, the messages are represented over a quaternary alphabet, including A, T, G, and C. The DNA one-time pad encryption mechanism is based on using substitution schemes governed by a long superpad consisting of a library of distinct subpads, where each subpad describes a randomly generated pair-wise mapping. More specifically, each subpad of the superpad consists of the WC complement of a plain-text sequence, the corresponding WC complement of the encrypted message, and a “stop” subsequence. The plain-text sequence acts as a primer of the transcription process that terminates with the stop sequence, similarly to what is known to occur during a regular cellular transcription cycle. The transcribed DNA message consists of the WC complement of the plain text and the WC complement of its encrypted sequence, so that the cypher text can be obtained by enzyme-based cleavage of the transcript at the plain text / cypher-text sequence boundary.

The substitution encryption method can also be used in conjunction with DNA microarrays. In this setting, a two-dimensional message is encoded in terms of an array of oligosequences placed at the spots of the chip. The sequences on the array are exposed to DNA solutions consisting of DNA encoding words of longer length than those placed on the array. The first parts of the encoding words represent the WC complements of the oligostrands at the microarray spots, while the second parts represent the encrypted versions of the individual chip oligosequences. After the encoding sequences are hybridized with their counterparts on the chip, the unhybridized parts of the encoding sequences are cleaved from their base strands and used to represent the cypher text of the two-dimensional DNA data array.

One can also devise DNA cryptosystems in terms of implementing them in DNA-based logic [41, 43]. For example, XOR DNA circuits used in one-time pad systems can be made to operate on the binary representations of DNA messages, therefore generating encrypted sequences. These circuits can be realized in terms of tiling models, following the simple sticky-end construction principles described in Section 12.4.2. Similarly, DNA transistors based on carbon nanotubes can be used for transforming both one- and two-dimensional DNA messages according to a key sequence.

DNA technology can also be used to develop steganography systems. In a DNA-based steganography system, preprocessed DNA messages are secretly tagged and mixed with unencoded messages. The key for decoding is the WC complementary sequence of the tag, which hybridizes with the tag and allows for identifying the messages. DNA sequence preprocessing is a necessary step in the encryption process, since otherwise the system may be vulnerable to security attacks [42]. The tagging process used in DNA steganography may also be used for other applications, including DNA barcoding [42].

In conclusion, it is interesting to note that DNA computers were also proposed for breaking known cryptographic systems.³ In [44, 45], it was shown that by using molecular computers one can compromise the security of the DES (Data Encryption Standard) scheme. As reported in [44, 45], such an attack can be accomplished in terms of using less than a gram of DNA, and by performing computations on a standard commercial computer. The attack can also be shown to be robust under errors, and it operates on the principles of the plain-text–cipher-text attack. For such an attack, it is assumed that the cryptanalyst has access to a plain-text message and its corresponding encoding. Key recovery is accomplished by a simple parallel brute-force strategy, largely facilitated by selective and efficient hybridization properties of WC-complementary sequences.

12.4.4 DNA/RNA-Aided Nanoparticle Assembly

Traditionally, nanoparticles are synthesized by using polymers that play the role of both catalysts and primers of the reaction. Recently, it was recognized that DNA and RNA sequences can be versatile tools in the process of mediating the growth of nanoparticles [46–51], capable of performing the same role as polymers. This property of DNA and RNA strands is facilitated by their specific recognition capability,

3. The results in [44, 45] suggest that, in principle, one could accomplish such a task under ideal experimental conditions. The real potential of the scheme has to be assessed experimentally.

governed by the WC hybridization rules, and their characteristic folding properties. The first step in recognizing RNA folding patterns useful for controlling particle growth consists in constructing large pools of random-like RNA sequences that include metal coordination sites. From this pool, only those strands that fold in a way that allows for controlled binding of particles (such as, for example, palladium particles [46]) are selected. The selection process is iterated several times in order to identify near-optimal folding shapes capable of binding to the already formed nanoparticle structure.

RNA strands usually fold into formations containing a number of loops and stem regions that allow for suitable interactions with other folds and particles. Self-assembly of nanoparticles is mostly achieved in terms of using RNA loops as “interacting” units between different RNA molecules that congregate in two- and/or three-dimensional structures. Self-assembly can be facilitated by templates or it can proceed without the use of templates. In the first case, assembly occurs under the direct guidance of some external force or external reaction-initiating sequences, based on certain predesigned hybridization constraints imposed on the RNA strands that enable it to respond to changes in the external force. In the second case, enzymes controlling ligation or linkage are used instead of external stimuli. Among the best-known RNA forms with applications in nanoparticle assembly is the *motor pRNA of bacteriophage* that can be easily altered to form complex three-dimensional shapes in terms of carefully reverse-engineered intertwined loop and stem structures. Usually, such properties are designed and controlled on the level of the primary sequence composition by incorporation of palindromic subsequences and inverted repeats [48]. In addition to their use for nanoparticle assembly, pRNAs are also gaining widespread application in therapeutic nanosystems that operate based on RNA interference principles and that involve various classes of ribozymes. In such systems, pRNAs can be shown to be able to control proliferation and apoptosis (death) of cancer cells [49–51].

12.5 DNA/RNA Sequence Hybridization: A Coding-Theoretic Point of View

This section contains an overview of techniques used to increase the efficiency and accuracy of DNA/RNA based systems using hybridization and self-hybridization processes. These techniques are mathematical in nature, and they rely on the vast knowledge regarding coding schemes used for data transmission and storage.

12.5.1 DNA Codes

One of the major problems encountered during the design of reliable DNA computers, DNA microarrays, and RNA-based nanosystems is the low reliability of DNA/RNA hybridization and enzyme-based operations. DNA computing experiments require the creation of a controlled environment that allows for a set of DNA oligonucleotide strands (in this context termed *codewords*) to hybridize with their complements in an appropriate fashion. If the codewords are not carefully chosen, unwanted, or nonselective, hybridization may occur. Furthermore, during

DNA computing and microarray hybridization experiments, oligostrands can form secondary and tertiary structures that render the sequences inactive. In addition to such reliability problems, for the purpose of designing smart drug systems, it may be necessary to determine what the characteristic RNA signature patterns of cancer cells are and how to govern the appropriate enzymatic reactions. Although it seems like an impossibly hard task to determine these patterns, tremendous progress on this front was made through the use of DNA microarray technology [38] and its accompanying analysis techniques borrowed from dynamical systems theory. One of the major insights gained through the use of microarray data is that any dynamic, biochemical information flow inside the cell can be described through time-varying expression patterns of genes; the product of an activated gene interacts with a variety of other biomolecules, gene products themselves, which in turn regulate the expression of some other genes through complex signaling mechanisms. Such a signaling loop is referred to as a *gene regulatory network* (GRN). Simply put, every process in a cell is governed by the interaction patterns between genes that act like multilevel switches. The activity level of these switches is measured in terms of the speed of gene transcription. By measuring the expression levels of genes involved, through the concentration of RNA molecules, one can obtain a data set known as the “RNA fingerprint.”

We start with the problem of designing DNA and RNA oligostrands with good hybridization and folding properties. Without loss of generalities, such strands will be henceforth viewed as words over a four-letter alphabet $Q = \{A, C, G, T\}$, and denoted by $q = q_1 q_2 \dots q_n$, where $q_i \in Q$ and where n indicates the length of the sequences. The sequence obtained by reversing q , that is, the sequence $q_n q_{n-1} \dots q_1$, will be denoted by q^R . The WC complement, or reverse-complement, of q is the sequence $q^{RC} = \bar{q}_n \bar{q}_{n-1} \dots \bar{q}_1$, where $\bar{q}_i$ denotes the WC complement of q_i as defined in Section 12.2.

$$\bar{A} = T, \bar{G} = C, \bar{T} = A, \bar{C} = G. \quad 12.1$$

The GC content of a DNA codeword $q = q_1 q_2 \dots q_n$ is the number of positions i such that q_i is in $\{C, G\}$. For any pair of length- n words, $q = q_1 q_2 \dots q_n$ and $p = p_1 p_2 \dots p_n$, over the alphabet Q , the Hamming distance $d_H(p, q)$ is defined as the number of positions in which the words differ. Furthermore, the *reverse Hamming distance* and the *reverse-complement Hamming distance* between the words p and q are defined as

$$\begin{aligned} d_H^R(p, q) &= d_H(p, q^R) \\ d_H^{RC}(p, q) &= d_H(p, q^{RC}) \end{aligned} \quad 12.2$$

respectively.

For a set of DNA codewords (i.e. a DNA code C) its minimum (Hamming) distance and its minimum reverse-complement (Hamming) are defined as the smallest Hamming and reverse-complement Hamming distance between any pair of non-identical codewords. If C is a DNA code in which all codewords have the same GC content, w , then C is called a constant GC content code, and w is termed the GC content of the code.

The distance measures defined above come into play when evaluating hybridization properties of DNA words under the assumption of a *perfectly rigid sugar-phosphate DNA backbone*. As an example, consider two DNA codewords,

3'-AAGCTA-5' and 3'-ATGCTA-5', at Hamming distance one from each other. For such a pair of codewords, the reverse complement of the first codeword, namely 3'-TAGCTT-5', will show a very large affinity to hybridize with the second codeword. In order to prevent such a possibility, one should impose a minimum Hamming distance constraint, $d_{\min}$, on the set of DNA codewords, for some sufficiently large value of $d_{\min}$. On the other hand, in order to prevent unwanted hybridization between two DNA codewords, one has to ensure that the reverse-complement distance between all codewords is larger than a prescribed threshold, say, $d_{\min}^{\text{RC}}$. Indeed, if the reverse-complement distance between two codewords is small, as for example in the case of the DNA strands 3'-AAGCTA-5' and 3'-TACCTT-5', then there is a good chance that the two strands will hybridize (incompletely).

Hamming distance is not the only measure that can be used to assess DNA cross-hybridization patterns. For example, if the DNA sugar-phosphate backbone is taken to be a perfectly elastic structure, then it is possible for bases not necessarily at the same position in two strands to pair with each other. For example, consider the two sequences:

$$\begin{aligned} &3'\text{-A(1,1)A(1,2)C(1,1)C(1,2)A(1,3)G(1,1)A(1,4)A(1,5)-5'} \\ &3'\text{-G(2,3)G(2,2)T(2,3)T(2,2)A(2,1)G(2,2)G(2,1)T(2,1)-5'} \end{aligned} \quad 12.3$$

Under the “perfectly elastic backbone” model, hybridization between the substrings of not necessarily consecutive bases

$$\text{A(1,2)C(1,1)C(1,2)A(1,3)A(1,4)} \quad 12.4$$

in the first strand, and

$$\text{T(2,1)G(2,1)G(2,2)T(2,2)T(2,3)} \quad 12.5$$

in the second strand, is plausible. The relevant distance measure for this model is the *Levenshtein distance* [52], which for a pair of sequences, p and q , is defined to be smallest number, $d_L(p; q)$, of insertions and deletions needed to convert p to q . A study of DNA codes with respect to a metric closely related to the Levenshtein metric can be found in [53]. The recent work of D'yachkov et al. [54, 55] considers a distance measure that is a slight variation on the Levenshtein metric and seems to be better in the DNA coding context than the Levenshtein metrics.

The *secondary structure* of an DNA codeword, $q = q_1q_2\dots q_n$ can be formally defined in the following broad sense. A secondary structure of size n is a graph on a set of n labeled points $\{1, 2, \dots, n\}$ such that the adjacency matrix $A = (a_{ij})$ of the graph has the following three properties:

1. $a_{i,i+1} = 1, 1 \leq i \leq n-1$
2. $i : \exists$ at most one $j \neq i \pm 1, \therefore a_{i,j} = 1$
3. $a_{i,j} = a_{k,l} = 1, i < k < j \Rightarrow i \leq l \leq j$

Here, nodes of the graph represent bases of the codeword, while edges specify base pairings. Folding usually occurs when the reverse-complement of a long subsequence of a codeword also appears in the codeword. Sometimes, folding is also due to the presence of tertiary structures known as *pseudoknots*, corresponding to pairings (q_i, q_j) and (q_k, q_l) for some q_i, q_j, q_k, q_l , such that $i < k < j < l$. Determining

if an oligonucleotide sequence will form a pseudoknot at a given temperature is known to be an NP-hard problem [56].

Based on the previous discussion, hybridization and secondary structure constraints imposed on the codewords used for DNA computing can be broadly classified into two groups, *individual* and *joint*. The first set of constraints is imposed on individual DNA sequences, and among the most important of these constraints are:

1. *The consecutive-bases constraint.* For certain applications, runs of the same base lead to an increase of hybridization errors; this introduces the need for a base-run constraint.
2. *The constant GC-content constraint.* The constant GC-content constraint is introduced in order to achieve parallelized operations on DNA sequences, by assuring similar thermodynamic characteristics of all codewords. These characteristics heavily depend on the GC content of the words, which is usually taken to be in the range of 30–50% of the length of the code.
3. *The secondary structure constraint.* This is realistically a whole class of constraints that are quite difficult to handle. Especially undesirable in this context are long runs of the pair GC, as well as long runs of both G and C. This is due to the fact that the chemical bond between G and C is stronger than that between A and T. Also important are the *stem length constraint* (the longer the stem, the stronger the overall stability of the fold), the *hairpin loop-length constraint* (formation of loops with more than 10 or less than 4 bases requires more energy), and the *unpaired base constraint* (unpaired bases decrease the stability of the structure).

Joint codeword constraints involve two or more codewords in a DNA code, and can be classified as follows:

1. *The Hamming distance constraint.* As already explained, it limits unwanted hybridizations between different codewords of the code.
2. *The reverse-complement Hamming distance constraint.* It limits undesired hybridization between a codeword and the reverse-complement of any other codeword.
3. *The frame-shift constraint* applies only to a limited number of computational problems. It refers to the requirement that the concatenation of two or more codewords should not properly contain any other codeword.
4. *The forbidden subsequence constraint.* This constraint specifies that a class of substrings must not occur in any codeword or concatenation of codewords, when, for example, restriction enzymes need to be used during the computation process and the binding sites must be clearly recognizable.

A comprehensive treatment of the subject of designing codes for DNA computing can be found in [55–57]. A sampling of the methodology used for constructing codes involving only the second individual and first joint constraint is presented below.

Method I: The construction of codes for DNA computing can be based on *reversible codes*. A collection of codewords C is said to be reversible if $c \in C$ implies that $c^R \in C$ [58]. Given a reversible code C over the finite field $GF(4)$, with minimum distance d , one can construct a DNA code by first eliminating all self-reversible codewords (i.e., codewords c for which $c^R = c$), and then choosing half of

the remaining codewords such that no codeword and its reverse are selected simultaneously. The code described above can be “transformed” into a DNA code by identifying the elements of $GF(4)$ with the DNA alphabet Q , and by replacing each of the first $\lfloor n/2 \rfloor$ symbols of $c \in C$ by their WC complement.

Method II: DNA codes can also be constructed from complex Hadamard matrices and constant-composition codes [58]. A *generalized Hadamard matrix* $H(n, C_m)$ is an $n \times n$ matrix with entries taken from the set of m^{th} roots of unity, $C_m = \{e^{-2i\pi l/m}, l = 0, \dots, m-1\}$, that satisfies $HH^* = nI$. Here, I denotes the identity matrix of order n , while $*$ stands for complex-conjugation. The *exponent matrix*, $E(n, Z_p)$, of $H(n, C_p)$ is a $n \times n$ matrix with entries in $Z_p = \{0, 1, 2, \dots, p-1\}$, for a prime p , obtained by replacing each exponential function in $H(n, C_p)$ by its exponent. A generalized Hadamard matrix is said to be in *standard form* if its first row and column consist of ones only. The $(n-1) \times (n-1)$ square matrix formed by the remaining entries of H is called the *core* of H , and the corresponding submatrix of the exponent matrix E is called the core of E . One can show that there exist complex Hadamard matrices with a core that is a circulant matrix consisting of all the $p^k - 1$ cyclic shifts of its first row. Such a core is referred to as a *cyclic core*. Each element of Z_p appears in each row of E exactly p^{k-1} times, and the Hamming distance between any two rows is exactly $(p-1)p^{k-1}$ [53]. Thus, the rows of the core of E form a code with constant composition (i.e., with a uniform distribution for the symbols). DNA codes with constant GC content can obviously be constructed from constant-composition codes over Z_4 by mapping the symbols of Z_4 to the symbols of the DNA alphabet, Q .

Method III: The DNA code design problem can also be reduced to a binary code design problem by mapping the DNA alphabet onto the set of length-two binary words as follows:

$$A \rightarrow 00, T \rightarrow 01, C \rightarrow 10, G \rightarrow 11 \quad 12.6$$

The mapping is chosen so that the first bit of the binary image of a base uniquely determines the complementary pair to which it belongs. The binary representation of the codewords can be de-interleaved so as to form the *even* and *odd* subsequences, which can then be designed separately. For the even component, codes with constant Hamming weights are used, while for the odd component, cyclic reversible codes can be used instead [53].

Several alternative approaches for designing DNA words with good hybridization properties based on the Hamming and Levenshtein distance measure were proposed in [59]. Among them, the most frequently used code structures are λ -free codes and c-h codes.

A set of DNA codewords is said to be λ -free if and only if no two codewords have a common substring of length greater than λ ; λ -free codes can be constructed from deBruijn sequences [59] and the optimal size of such codes can be shown to be

$$4^\lambda / (n - \lambda + 1) \quad 12.7$$

A set of DNA codewords comprises a c-h code provided that each codeword has *nucleic weight* lower bounded by h , and all the subsequence (of codewords) with nucleic weight lower bounded by c are allowed to occur at most once in the code. The nucleic weight of a DNA codeword is defined as the weighted sum

$$\sigma_c(A, T) + 3\sigma_c(G, C), \quad 12.8$$

with $\sigma_c(A, T)$, $\sigma_c(G, C)$ denoting the number of A,T and G,C bases in the DNA codeword respectively.

On the other hand, determining the *exact pairings* in a secondary DNA structure is a very complicated task due to the following problems. For a system of interacting entities, one measure commonly used for assessing the system's property is the *free energy*, which is also one of the fundamental quantities in statistical mechanics [61]. It equals the average energy of the system from which the temperature-normalized system entropy is subtracted. Stable states of systems usually correspond to states of minimum free energy. The energy of a base pairing in a DNA/RNA secondary structure depends on the bases involved as well as all adjacent bases. Furthermore, in the presence of other neighboring pairings, these energies change based on some nontrivial rules. Nevertheless, some simple dynamic programming techniques can be used to *approximately* predict the minimum free-energy secondary structure of a DNA sequence without taking pseudoknots into account. Among these techniques, *Nussinov's folding algorithm* is the most widely used scheme [60]. Nussinov's algorithm is based on the following simple assumptions.

Let $F_{i,j}$ be the minimum free energy of a DNA subsequence $c_i \dots c_j$. Assume that the energy between a pair of bases $\alpha(c_k, c_l)$ is independent of all other pairs and let $\alpha(c_k, c_l) = \alpha < 0$ if c_k, c_l are WC complementary, and zero otherwise. For simplicity, one can assume that $\alpha = -1$, although other values of the parameter, depending on the choice of the base pair, can be used as well. The free energy of the sequence $c_i \dots c_j$ can then be found according to the formula

$$F_{i,j} = \min \begin{cases} F_{i+1,j-1} + \alpha(c_i, c_j) \\ F_{i,k-1} + F_{k,j} & i < k \leq j \end{cases} \quad 12.9$$

where $F_{i,j} = 0$ for $i = 1 \dots n$ and $j \leq i$. The value of $F_{1,n}$ is the minimum free energy of the secondary structure of $c_1 \dots c_j$, while the structure itself can be found by the *backtracking algorithm* [60]. A very low negative value for the free energy $F_{1,n}$ of a sequence indicates the presence of stems and loops.

Nussinov's algorithm is part of many commercial secondary structure prediction programs, such as the *Vienna secondary structure package* [62]. The Vienna package uses more elaborate versions of Nussinov's dynamic programming approach, as well as more accurate values of the parameters $\alpha(c_i, c_j)$. Nussinov's algorithm can also be visualized in terms of free-energy tables, examples of which are shown in Figure 12.8. In such tables, the entry at position (i, j) contains the minimum free energy of the subsequence $c_i \dots c_j$. The table is filled out by initializing the entries at the main diagonal and in the lower triangle to zero and calculating the energy levels according to (12.9). Observe that the value at position (i, j) , $j > i$, depends on $\alpha(c_i, c_j)$ as well as the values of all entries in (i, l) , $l = 1 \dots j - 1$, (l, j) , $l = i + 1 \dots n - 1$.

If a DNA code consists of cyclic shifts of one codeword only, as is the case with codewords constructed according to Method II described above, the computation of the minimum free energy of the codewords simplifies significantly. It can be shown that, provided that the free energy table of a DNA codeword $c_1 \dots c_n$ is known, the free-energy tables of *all other codewords* can be computed with a total of $O(n^3)$ operations only. More precisely, the free-energy table of the codeword $c_n, c_1 \dots c_{n-1}$ can be obtained from the table of $c_1 \dots c_n$ in $O(n^2)$ steps. For example, based on the free-energy table of the DNA sequence CCCAAATGG shown in Figure 12.8, the free-

	C	C	C	A	A	A	T	G	G		G	C	C	C	A	A	A	T	G
C	<u>0</u>	0	0	0	0	0	-1	-2	-3	G	<u>0</u>	-1	-1	-1	-1	-1	-1	-2	-3
C	<u>0</u>	<u>0</u>	0	0	0	0	-1	-2	-3	C	<u>0</u>	<u>0</u>	0	0	0	0	0	-1	-2
C	*	<u>0</u>	<u>0</u>	0	0	0	-1	-2	-2	C <td>*</td> <td><u>0</u></td> <td><u>0</u></td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>-1</td> <td>-2</td>	*	<u>0</u>	<u>0</u>	0	0	0	0	-1	-2
A	*	*	<u>0</u>	<u>0</u>	0	0	-1	-1	-1	C <td>*</td> <td><u>0</u></td> <td><u>0</u></td> <td><u>0</u></td> <td>0</td> <td>0</td> <td>0</td> <td>-1</td> <td>-2</td>	*	<u>0</u>	<u>0</u>	<u>0</u>	0	0	0	-1	-2
A	*	*	*	<u>0</u>	<u>0</u>	0	-1	-1	-1	A <td>*</td> <td>*</td> <td><u>0</u></td> <td><u>0</u></td> <td><u>0</u></td> <td>0</td> <td>0</td> <td>-1</td> <td>-1</td>	*	*	<u>0</u>	<u>0</u>	<u>0</u>	0	0	-1	-1
A	*	*	*	*	<u>0</u>	<u>0</u>	-1	-1	-1	A <td>*</td> <td>*</td> <td>*</td> <td><u>0</u></td> <td><u>0</u></td> <td><u>0</u></td> <td>0</td> <td>-1</td> <td>-1</td>	*	*	*	<u>0</u>	<u>0</u>	<u>0</u>	0	-1	-1
T	*	*	*	*	*	<u>0</u>	<u>0</u>	0	0	A <td>*</td> <td>*</td> <td>*</td> <td>*</td> <td><u>0</u></td> <td><u>0</u></td> <td><u>0</u></td> <td>-1</td> <td>-1</td>	*	*	*	*	<u>0</u>	<u>0</u>	<u>0</u>	-1	-1
G	*	*	*	*	*	*	<u>0</u>	<u>0</u>	0	T <td>*</td> <td>*</td> <td>*</td> <td>*</td> <td>*</td> <td><u>0</u></td> <td><u>0</u></td> <td><u>0</u></td> <td>0</td>	*	*	*	*	*	<u>0</u>	<u>0</u>	<u>0</u>	0
G	*	*	*	*	*	*	*	<u>0</u>	<u>0</u>	G <td>*</td> <td>*</td> <td>*</td> <td>*</td> <td>*</td> <td>*</td> <td><u>0</u></td> <td><u>0</u></td> <td><u>0</u></td>	*	*	*	*	*	*	<u>0</u>	<u>0</u>	<u>0</u>

Figure 12.8 An illustration of the Nussinov folding algorithm for a DNA sequence and its cyclic shifts.

energy table of the codeword GCCCAAATG can be found by filling out only the first row. This approach provides for significant computational savings.

12.5.2 DNA Microarrays

Microarrays are another example of DNA-based systems that can exhibit large error rates during operation due to improper hybridization issues. The probe selection process for these arrays usually follows the design criteria outlined in the previous section, with certain additional constraints imposed on the sequence content. These constraints arise from the fact that probes cannot be arbitrarily selected, but must be properly chosen subsequences of natural DNA coding regions. More details regarding the probe selection process can be found in [53, 57].

In this section, the focus is on coding techniques used to improve the precision of the manufacturing process of DNA microarrays. Errors in the read-out signal can be attributed not only to improper hybridization, but also to missed steps in the production process, substrate defects, malfunctioning of the optical detection system, background illumination, and other phenomena. In order to ensure high data integrity, Affymetrix [37] introduced into its production process a sophisticated technology that utilizes designated quality control spots for detecting erroneously synthesized DNA strands. Although these procedures ensure that DNA arrays are properly created, they provide no built-in error-control mechanism that allows an array to recover from subsequent loss of data in several spots in the grid.⁴ This motivated the authors of [63] to propose a DNA strand multiplexing scheme with redundant spots that ensures reliable operation of the array in the presence of multiple spot failures. The results in [63] were validated in terms of fabricating chips of small size. Since the tests were performed on arrays with six, eight, and ten spots only, the problems of quality control and cost of the production process were not addressed.

Three important mathematical problems can be posed regarding the choice of the base schedule, the mask structures, and the possibility of detecting production failures in a DNA microarray:

1. *Base Scheduling*: One problem of interest is to find the shortest possible asynchronous schedule that can produce a set of predefined probes. The shortest schedule ensures reductions in the cost of chip production and also

4. It is worth pointing out that the Affymetrix chip has a built-in redundancy, in the sense that several spots are reserved for each DNA probe.

decreases the overall error probability of probe formation. It is straightforward to see that such a schedule takes the form of a shortest common superstring of the probes, the computation of which is known to be NP-hard. Furthermore, since the probes are usually selected in such a way that they poorly hybridize with each other, no large probe sequence overlaps are expected. Consequently, most currently available fabrication methods use synchronous schedules.

2. *Mask Design*: Since spots are activated by illumination, it is possible that—due to reflections, imprecision in the optical system, and the mask layout—spots neighboring the target spot become exposed to light as well. This unintended illumination can activate linkers of spots that were supposed to be masked, resulting in erroneous probe synthesis. One method to mitigate these effects is to minimize the total length of the borders of masks used in the VLSIPS process [64–66]. The total border length of masks under a synchronous schedule corresponds to the sum of all pair-wise Hamming distances between probes at adjacent spots (adjacency in this context means that spots share at least one “edge”). If the set of probes consists of all possible DNA sequences of a given length, the minimum border length is achieved in terms of using two-dimensional Gray code masks [67]. When the probes are structurally constrained, some of the best known mask design methods rely on constructing *probe neighborhood graphs* [65, 66]. Neighborhood graphs are complete graphs with vertices corresponding to the probes; the weight of an edge bridging two vertices equals the Hamming distance between the probes corresponding to the vertices. For such a graph, an approximate solution for the traveling salesman (TS) problem is sought, which specifies an ordering of the probes that is to be transferred to the array. This transfer is achieved in terms of *threading*, based on the use of certain discretized space-filling curves that embed a one-dimensional string into a two-dimensional grid [66, 67].
3. *Quality Control*: As described in the *Manufacturing Quality Control and Validation Studies of GeneChip Arrays Manual* in [38] the accuracy of the production steps is controlled in terms of a designated subset of *quality control spots* [65, 66]. In this setting, m quality control spots contain $c < m$ different probes, and identical probes are synthesized using different steps in the manufacturing process. By performing hybridization tests on quality control spots one can identify if a faulty step occurred during probe synthesis. The work in [68, 69] extends this testing framework one step further, by proposing a coding scheme for identifying one erroneous production step. Information about erroneous production steps can be used to discover systematic problems in the production process.

From the perspective of coding theory the quality control problem can be formulated by referring to the following combinatorial objects. Let A be an $M \times N$ array of binary numbers such that the weight of each row equals r , the Hamming distance between any two columns is at least d , and the weight of each column is within the interval $[w_{\min}, w_{\max}]$, where $w_{\min} < w_{\max}$. Then A is called a *balanced binary code* with parameters $(M; N; r; d; w_{\min}; w_{\max})$. A balanced binary code can be in-

interpreted as an array of M quality control spots and N production steps. A “1” is placed at position $(i; j)$ of the array if and only if the i^{th} quality control spot was active during the j^{th} step of the fabrication process. The constant row weight constraint is imposed in order to ensure that all quality control probes are of the same length, while the Hamming distance guarantees that two productions steps have distinguishable signatures even in the presence of d spot dropouts. The restrictions on the weights of the columns guarantee that one can distinguish between a step failing and a step not being used during the production process, since for hybridization experiments one usually measures relative rather than absolute fluorescence intensities.

The problem of designing balanced codes is well understood. To construct arrays corresponding to balanced codes it suffices to use combinatorial designs or subsets of codewords of codes with large minimum distance [68, 69]. Far more interesting is the problem of detecting multiple production step failures. This question can be addressed in the framework of *superimposed designs* described in [70]. Let A be an $M \times N$ array of binary numbers with columns $x_1, x_2, \dots, x_N$ and rows of weight r . The array A is said to be a $(M; N; r; s)$ -*superimposed design* with constraint r and strength s if all component-wise Boolean OR functions of not more than s columns of A are distinct. Assume now that the quality control matrix of a microarray corresponds to a superimposed design with constraint r . Then each probe has length r and multiple production step failures can be detected as follows. First, note that in order for a probe to be erroneously synthesized, *at least one* production step during which the probe was active has to fail. Consequently, the hybridization intensity profile of quality control probes contains information about the *component-wise Boolean OR function* of the columns of the control array. If each Boolean OR function of not more than s columns is unique, one can identify any set of not more than s failed production steps. It is important to observe that there is no guarantee that a quality control scheme based on a superimposed design can guarantee proper identification of multiple production step failures in the presence of spot dropouts in the control array.

Upon completion of the DNA microarray fabrication and testing process, the structure and properties of the array can change so that certain spots become non-functional. Spot failure is a common event that is very hard to detect since it manifests itself in terms of low fluorescence intensity during a hybridization experiment. But low fluorescence at a given spot can also be attributed to the gene corresponding to the synthesized probe being inactive in the tested cell. Consequently, there exists a strong need to design microarrays in such a way that, even under spot failure, information about every single gene originally present in the grid is available. This can be achieved by multiplexing identical probes to different spots in the array [63]. In what is henceforth called a *multiplexed array*, every spot contains a mixture of a fixed number of different probes, and the number of spots exceeds the number of tested probes. For an array with the aforementioned properties, one seeks to design a binary *multiplexing matrix*, G , of dimension $M \times N$, where M denotes the number of spots, N denotes the number of distinct probes, and $M > N$. The matrix G has to have full rank and the property that $G(i, j) = 1$ if and only if the i^{th} spot contains the j^{th} probe. Under the assumption that all spots have identical properties, that the system noise is additive and i.i.d., and that all probes show identical hybridization affinities with their complements, the optimal choice for G is the one that minimizes

$$\text{tr}(G^{*T} G^*) \quad 12.10$$

where G^* denotes the pseudo-inverse of G defined as $G^* = (G^T G)^{-1} G^T$. An example of a multiplexing matrix with $M = 6$ and $N = 4$, found by computer search in [63], is shown in (12.11):

$$G^T = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad 12.11$$

For the above example, in the presence of any two spots failures the information about gene probes remains intact for comparative study. Each spot contains two probes and each probe appears at three different spots. Note that the number of different probes placed at each spot should be small, and that the probes synthesized at the same spot should poorly hybridize with each other and their complementary DNA sequences. For more details about coding schemes used for quality- and error-control coding in DNA microarrays, the interested reader is referred to [71].

12.5.3 Enumerating RNA Motifs

Aptamers and other functional RNA sequences are usually identified during extensive experiments that involve large pools of random-like genetic sequences. The success of the experiment depends to a large extent on the statistical setup of the testing scheme [72–74] and the choice of the RNA sequences used in the pool. The number of random RNA sequences used in the experiment has to be large enough to guarantee a nonnegligible frequency of occurrence of all possible RNA folding patterns in the test tube. This introduces the need to analytically enumerate all possible RNA folding motifs and compute their probabilities, since for short RNA strands counting all motifs is expensive, and for long RNA strands counting all folding motifs is both experimentally and computationally intractable.

Several combinatorial approaches were suggested for counting specific RNA folding structures. Waterman [75] proposed a method for counting different hairpin and cloverleaf patterns in random RNA sequences, based on some simplified assumptions regarding the structure of the RNA sugar-phosphate backbone. Viennot [76] considered a set of simple transformations that map RNA secondary structures into extended binary trees and Dyck paths, and enumerated all possible binary tree structures that can arise in this manner. He also considered the Horton-Strahler (HS) number of the underlying tree representations, describing the “branching complexity” of a secondary structure, also known as its *order*. The HS number of a binary tree is defined recursively, by labeling the leaves in the tree with “1.” The label of the parent of two nodes is determined according to the following rule: if both children have the same label l , the label of the parent node is set to $l + 1$ [77]. Otherwise, the label of the parent is the maximum of the labels of its children. The Horton-Strahler number equals the label of the root of the tree. An example illustrating the process of computing the HS number of a tree is shown in Figure 12.9.

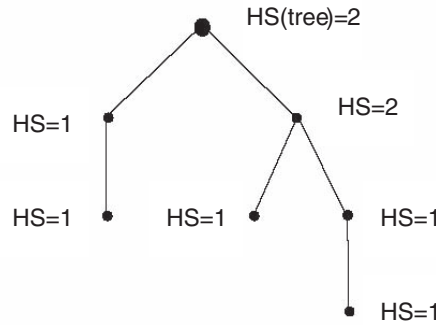


Figure 12.9 Computation of the Horton-Strahler number of a tree.

Regnier [78] computed the generating function $S^{[h,b]}(z)$ of the number of secondary structures of RNA strands of length n , with minimal stem and loop size h and b respectively, and showed that it satisfies the following recursion

$$(S^{[h,b]}(z))^2 z^{2h} + S^{[h,b]}(z) \left[(z-1)(1-z^2+z^{2h}) - z^{2h} \frac{1-z^b}{1-z} \right] + 1 - z^2 + z^{2h} = 0 \quad 12.12$$

An asymptotic expression for the number of secondary structures of given length n can be determined from a classical asymptotic analysis of (12.12).

In order to illustrate the above described concepts and counting arguments, let us consider the DNA folding pattern shown in Figure 12.10.

The secondary structure of the sequence 3'-AGGCTAAAAGCCT-5' is mapped into a one-dimensional sequence consisting of the symbols “(”, “)”, and “|”. The conversion is performed in the following manner. First, a reading direction for the RNA strand is fixed. Without loss of generality, this can be the 3'-5' direction of the strand. A base is represented by the symbol “(” if it corresponds to a paired base that is the first one encountered while scanning the sequence. Similarly, a base is represented by the symbol “)” if it is the second base in the pairing encountered in the 3'-5' direction. The symbol “|” is used to represent unpaired bases.

Not every sequence over the alphabet $\{ (,), | \}$ corresponds to a valid DNA or RNA secondary structure. There are several constraints that can be easily identified as necessary conditions for a sequence to correspond to a valid folding pattern. First, the number of first bases in a collection of base pairs always has to be larger than or equal to the number of second bases. This implies that all the prefixes of a word representing a DNA/RNA fold have to have the property that they contain more “(” than “)” symbols, or an equal number of “(” and “)” symbols. The complete sequence has to contain the same number of “(” and “)” symbols. Sequences of length n satisfying this property are known as Dyck's words or Dyck's paths [77] and they are counted by the well-known Catalan numbers, given by

$$C_n = \frac{1}{n+1} \binom{2n}{n} \quad 12.13$$

Dyck's words of limited maximal symbol disparity can be generated by walks through a graph known in coding theory as the *RDS (Running Digital Sum) graph*,

shown in Figure 12.10. In this graph, the states denote the difference in the number of “(” and “)” symbols in a word, counted up to a given position. Consequently, only states labeled with non-negative integers are included, and a transition from one state to another is possible if and only if the labels of the two states differ by 1 in absolute value (the initial and final state excluded). Dyke words are generated by following paths starting from state “0” and ending at state “0”. The number of such paths of a given length $2n$ can be found by representing the graph in terms of its adjacency matrix M , shown for the case of $N = 5$ states below, and computing the entry at position (1,1) in M^{2n} .

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad 12.14$$

More generally, the class of DNA and RNA folding patterns can be described by the so-called *Motzkin* and *Schroeder paths* [79]. A Motzkin path is a lattice path that uses the steps (1; 1), (1; -1), and (1; 0), and that begins and ends on the x axis and never goes below it. A Schroeder path is a lattice path that uses the steps (1; 1), (1; -1), and (2; 0), and that begins and ends on the x axis and never goes below it. These paths are counted by the well-known Motzkin numbers m_n and Schroeder numbers r_n with generating functions

$$\sum_{n \geq 0} m_n t^n = \frac{1 - t - \sqrt{1 - 2t - 3t^2}}{2t^2}$$

$$\sum_{n \geq 0} r_n t^n = \frac{1 - t - \sqrt{1 - 6t + t^2}}{2t} \quad 12.15$$

If the steps (1; 1) and (1, -1) are identified by the symbols “(” and “)” respectively, and the steps (2; 0) and (1; 0) are identified by the symbol “|”, then RNA/DNA folding structures can be mapped into Motzkin and Schroeder paths.

Another important constraint that has to be taken into account when using this counting method comes from the physical properties of the DNA sugar-phosphate backbone. The constraint is usually called the *no-sharp-turn constraint*, and it asserts that the size of any loop in a secondary or tertiary structure has to be at least three. This implies that all symbols “|” in a word describing a DNA or RNA fold

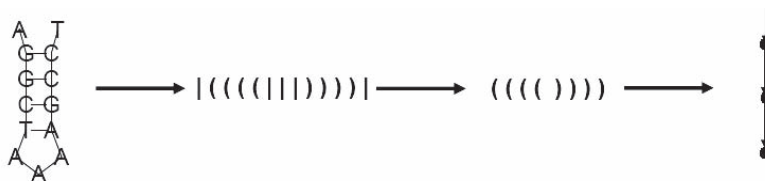


Figure 12.10 The secondary structure of the sequence 3'-AGGCTAAAAGCCT-5', its ternary symbol representation, and the corresponding tree structure.

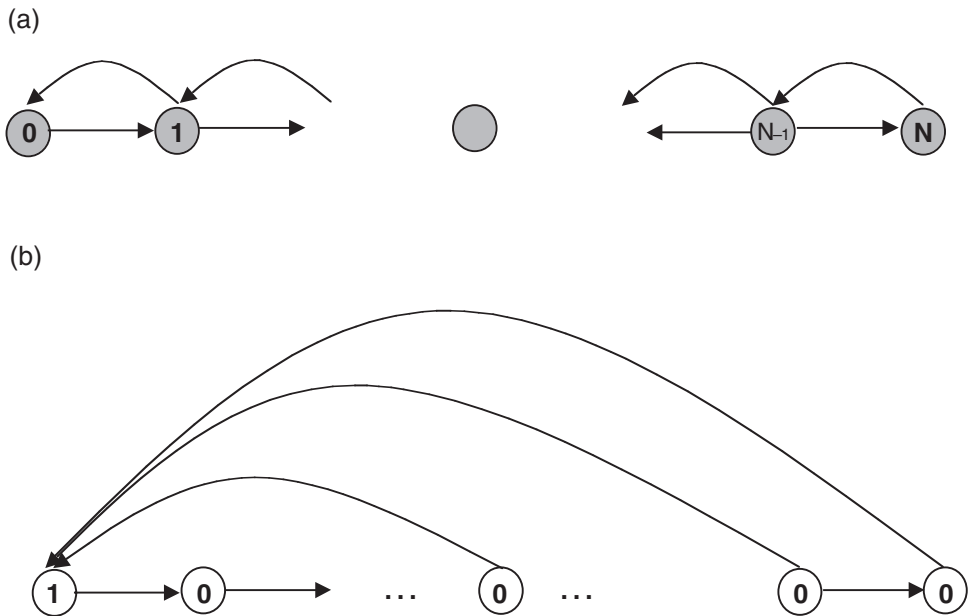


Figure 12.11 The RDS and RLL graph.

have to appear in groups of at least three elements. In coding theory literature, constraints of this form are known as *run-length constraints* (RLL constraints) [80]. Binary words satisfying a run-length constraint can be generated by following paths through an RLL graph, an example of which is shown in Figure 12.11. There, phrases of one of the symbols (say, zero) are restricted to have length in a limited set of values (larger than $D - 1$ and less than or equal to $N - 1$).

Restrictions of the above form can be combined to derive composite constraints. If the RDS constraint is imposed on the “(” and “)” symbols, and the RLL constraint is imposed on the “|” symbol, the sequences generated by the composite constrained graph can be used to model specific classes of RNA folding patterns. More details about these mapping techniques can be found in [81], while some alternative approaches to RNA/DNA motif enumeration are described in [82, 83].

12.6 Conclusion

DNA and RNA sequence hybridization and self-hybridization are two of the most important biological processes responsible for maintaining and controlling important cell functions. Various aspects of hybridization can also be exploited in man-made systems for the purpose of designing DNA computers, smart drugs, cryptographic systems, and self-assembly nanodevices. For such devices, operational reliability represents a major problem that can be addressed in terms of utilizing coding-theoretic techniques and principles developed for classical communication systems.

References

- [1] Strachen, T., and A. Read, *Human Molecular Genetics*, 3rd ed., New York: Garland Scientific Pub., 2003.
- [2] <http://nobelprize.org/chemistry/laureates/1989/press.html>
- [3] Serganov, A., et al., "Structural bias for gene regulation by a thiamine pyrophosphate-sensing riboswitch," *Nature*, May 2006 (on-line publication date).
- [4] Web resources: <http://www.actigenics.com/site>
- [5] L. Gold, et al., "From oligonucleotide shapes to genomic SELEX: Novel biological regulatory loops," *Proc. Natl. Acad. Sci. USA*, Vol. 7, No. 94, No. 1, Jan. 1997, pp. 59–64.
- [6] Web resources: ARCHEMIX-The aptamer therapeutics company, <http://www.archemix.com>
- [7] Famulok, M., and J. Szostak, "Selection of functional RNA and DNA molecules from randomized sequences," *Nucleic Acids Mol. Biol.*, Vol. 7, pp. 271, F. Eckstein, D. M. J. Lilley (eds.), Berlin: Springer Verlag, 1993.
- [8] Web resources: *Journal of RNAi and Gene Silencing*—An international journal of RNA and gene targeting research, <http://www.libpubmedia.co.uk/RNAiJ/>
- [9] Ovcharenko, D., et al., "High throughput RNAi screening in vitro: from cell lines of primary cells," *RNA*, Vol. 11, 2005, pp. 985–993.
- [10] Benan, M., and N. Puri, "The ins and out of RNAi in mammalian cells," *Curr. Pharm. Biotechnol.*, Vol. 5, No. 5, 2004, pp. 441–450.
- [11] *RNA Editing web-resource*: http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/R/RNA_Editing.html
- [12] Hoopengardner, B., "Nervous system targets of RNA editing identified by comparative genomics," *Science*, Vol. 301, 2002, pp. 832–836.
- [13] Clutterbuck, D. R., et al., "A bioinformatics screen for novel A-I RNA editing sites reveals recoding editing in BC10," *Bioinformatics*, Vol. 21, No. 11, 2005, pp. 2590–2595.
- [14] Rothman-Dense, L. B., et al., "Transcriptional regulation by DNA structural transitions and single-stranded DNA-binding proteins," *Cold Spring Harbor Symp. Quant. Biol.*, Vol. 63, 1998, pp. 63–73.
- [15] Paquin, B., M.-J. Laforest, and B. F. Lang, "Double-hairpin elements in the mitochondrial DNA of *Allomyces*: evidence for mobility," *Mol. Biol. Evol.*, Vol. 17, 2002, pp. 1760–1768.
- [16] Moore, H., et al., "Triplet repeats form secondary structures that escape DNA repair in yeast," *Proc. Natl. Acad. Sci. USA*, Vol. 96, 1999, pp. 1504–1509.
- [17] Jones, C., et al., "Co-localisation of CCG repeats and chromosome deletion breakpoints in Jacobsen syndrome: evidence for a common mechanism of chromosome breakage," *Hum. Mol. Genet.*, Vol. 9, No. 8, 2000, pp. 1201–1208.
- [18] Nag, D. K., M. Suri, and E. K. Stenson, "Both CAG repeats and inverted DNA repeats stimulate spontaneous unequal sister-chromatid exchange in *Saccharomyces cerevisiae*," *Nucleic Acids Res.*, Oxford Univ. Press, Vol. 32, No. 18, 2004, pp. 5677–5684.
- [19] Sinden, R. R., et al., "Triplet repeat DNA structures and human genetic disease: dynamic mutations from dynamic DNA," *J. Biosci.*, Vol. 27, 2002, pp. 53–65.
- [20] Kurahashi, H., and B. Emanuel, "Long AT-rich palindromes and the constitutional t(11;22) breakpoint," *Hum. Mol. Genet.*, Vol. 10, 2001, pp. 2605–2617.
- [21] Peng, Q., P. Pevzner, and G. Tesler, "The fragile breakage versus random breakage models of chromosome evolution," *PLoS Comput. Biol.*, Vol. 4, No. 2, Feb. 2006, p. e14.
- [22] Adleman, L., "Molecular computation of solutions to combinatorial problems," *Science*, Vol. 266, 1994, pp. 1021–1024.
- [23] Braich, R. S., et al., "Solution of a 20-variable 3-SAT problem on a DNA computer," *Science*, Vol. 296, Apr. 2002, pp. 492–502.

- [24] Faulhammer, D., et al., "Molecular computation: RNA solutions to chess problems," *Proc. Natl Acad. Sci. USA*, Vol. 97, No. 4, Feb. 2000, pp. 1385–1389.
- [25] Stojanovic, M. N., T. E. Mitchell, and D. Stefanovic, "Deoxyribozyme-based logic gates," *J. Amer. Chem. Soc.*, Vol. 124, No. 14, Apr. 2001, pp. 3555–3561.
- [26] Stojanovic, M. N., and D. Stefanovic, "Deoxyribosome-based half-adder," *J. Amer. Chem. Soc.*, Vol. 125, No. 22, 2003, pp. 6673–6676.
- [27] Sun, L.-Q., "Use of deoxyribozymes for gene knockdown," *Med. Chem. Rev. Online*, Vol. 2, No. 1, Feb. 2005, pp. 81–87.
- [28] Stojanovic, M. N., and D. Stefanovic, "DNA beats human challenger in tic-tac-toe," *Nat. Biotechnol.*, Sept. 2003, pp. 1069–1074.
- [29] Keren, K., et al., "DNA-templated nanotube field-effect transistor," *Science*, Vol. 302, No. 5648, Nov. 2003, pp. 1380–1382.
- [30] Mansuripur, M., et al., "Information storage and retrieval using macromolecules as storage media," University of Arizona Technical Report, 2003.
- [31] Roco, M., "Nanotechnology: convergence with modern biology and medicine," *Curr. Opin. Biotechnol.*, Vol. 14, 2003, pp. 337–346.
- [32] Tsaftaris, S., et al., "DNA computing from a signal processing viewpoint," *IEEE Signal Processing Magazine*, Sept. 2004, pp. 100–106.
- [33] Benenson, Y., et al., "An autonomous molecular computer for logical control of gene expression," *Nature*, Vol. 429, May 2004, pp. 423–429.
- [34] Seeman, N. N., "Nucleic acid nanostructures and topology," *Angew. Chem. Intl. Ed.*, Vol. 37, 1998, pp. 3220–3238.
- [35] Winfree, E., "DNA computing by self-assembly," *Proc. Natl. Acad. Eng.*, Vol. 33, No. 4, 2003, pp. 31–38.
- [36] Winfree, E., et al., "Design and self-assembly of two-dimensional DNA crystals," *Nature*, Vol. 394, 1998, pp. 539–544.
- [37] Wang, H., "Games, logic and computers," *Sci. Amer.*, Nov. 1965, pp. 98–106.
- [38] Affymetrix: Web resources at <http://www.affymetrix.com/>
- [39] Smith, K., "Universal microarrays: an algorithmic approach," web resources at <http://www.cs.mcgill.ca/~kaleigh/>
- [40] Dove, A., "DNA cryptography," *Nat. Biotechnol.*, Vol. 17, No. 7, p. 625, 1999.
- [41] Leier, A., et al., "Cryptography with binary DNA strands," *Biosystems*, Vol. 57, No. 1, June 2000, pp. 13–22.
- [42] Gehani, A., T. LaBean, and J. Reif, "DNA-Based Cryptography," *Proc. 5th DIMACS Workshop on DNA Based Computers*, pp. 167–188, Cambridge, MA: MIT, June 14–15, 1999.
- [43] Chen, J., "A DNA-based, biomolecular cryptography design," *Proc. Intl. Symp. Circuits and Systems, ISCAS '03*, Vol. 3, No. 25–28, May 2003, pp. 822–825.
- [44] Boneh, D., C. Dunworth, and R. Lipton, "Breaking DES using a molecular computer," Princeton CS Technical Report, CS-TR-489-95.
- [45] Adleman, L., et al., "On applying molecular computation to the data encryption standard," *Proc. Second Ann. Mtg. on DNA Based Computers*, Princeton Univ., June 10–12, 1996
- [46] Gugliotti, L., D. Feldheim, and B. Eaton, "RNA mediated metal-metal bond formation in the synthesis of hexagonal palladium nanoparticles," *Science*, Vol. 304, No. 5672, May 2004, pp. 850–852.
- [47] C. Niemeyer, "Nanotechnology: tools for the biomolecular engineer," *Science*, Vol. 297, No. 5578, July 2002, pp. 62–63.
- [48] Lukeman, P., N. C. Seeman, and A. Mittal, "Hybrid PNA/DNA nanosystems," in *First Intl. Conf. Nanoscale/Molecular Mechanics (N-M2-I)*, Outrigger Wailea Resort, Maui, Hawaii, 2002.

- [49] <http://www.chem.northwestern.edu/~mkngpr/BioNanomaterials2003rev1.htm>, *Nano Lett.*, Vol. 5, No. 9, 2005, pp. 1797–1808.
- [50] Guo, P., “RNA nanotechnology: engineering, assembly and applications in detection, gene delivery and therapy,” *J. Nanosci. Nanotechnol.*, Vol. 5, No. 12, Dec. 2005, pp. 1964–1982.
- [51] Khaled, A., et al., “Controllable self-assembly of nanoparticles for specific delivery of multiple therapeutic molecules to cancer cells using RNA nanotechnology,” *Amer. Chem. Soc.*, Sept. 2005, Web release.
- [52] Levenshtein, V. I., “Binary codes capable of correcting deletions, insertions, and reversals,” *Dokl. Akad. Nauk SSSR*, Vol. 163, No. 4, 1965, pp. 845–848 (in Russian). English translation in *Soviet Physics Doklady*, Vol. 10, No. 8, 1966, pp. 707–710.
- [53] Milenkovic, O., and N. Kashyap, “On the design of codes for DNA computers,” *Lecture Notes in Computer Science*, 3969, Berlin/Heidelberg: Springer Verlag, 2006, pp. 100–119.
- [54] D’yachkov, A., et al., “Exordium for DNA codes,” *J. Comb. Optim.*, Vol. 7, No. 4, 2003, pp. 369–379.
- [55] D’yachkov, A., et al., “New results on DNA codes,” *Proc. IEEE Int. Symp. Inform. Theory (ISIT ’05)*, Adelaide, Australia, Sept. 2005, pp. 283–287.
- [56] Marathe, A., A. E. Condon, and R. M. Corn, “On combinatorial DNA word design,” *J. Comput. Biol.*, Vol. 8, 2001, pp. 201–219.
- [57] Gaborit, P., and O. D. King, “Linear constructions for DNA codes,” *Theoret. Compu. Sci.*, Vol. 334, No. 1–3, Apr. 2005, pp. 99–113.
- [58] MacWilliams, F. J., and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam: North-Holland, 1977.
- [59] Morris, M., et al., “Methods and compositions for selecting tag nucleic acids and probe arrays,” European Patent Application 97302313, 1997.
- [60] Nussinov, R., and A. B. Jacobson, “Fast algorithms for predicting the secondary structure of single stranded RNA,” *Proc. Natl. Acad. Sci. USA*, Vol. 77, No. 11, 1980, pp. 6309–6313.
- [61] Zuker, M., “Mfold web server for nucleic acid folding and hybridization prediction,” *Nucleic Acids Res.*, Vol. 31, No. 13, 2003, pp. 3406–3415.
- [62] The Vienna RNA Package, <http://rna.tbi.univie.ac.at/cgi-bin/RNAfold.cgi>
- [63] Khan, A., et al., “Error-correcting microarray design,” *Genomics*, Vol. 81, 2003, pp. 157–165.
- [64] Hannehalli, S., et al., “Combinatorial algorithms for design of DNA arrays,” *Adv. Biochem. Eng./Biotechnol.*, Vol. 77, pp. 1–19, Springer Verlag, 2002.
- [65] Hubbell, E., and P. Pevzner, “Fidelity probes for DNA arrays,” in *Proc. 7th Intl. Conf. Intelligent Systems for Mol. Biol.*, Heidelberg, Germany, Aug. 1999, pp. 113–117.
- [66] Kahng, A., et al., “Design flow enhancement for DNA arrays,” *Proc. 21st Intl. Conf. Computer Design (ICCD ’03)*, 2003.
- [67] Feldman, W., and P. Pevzner, “Gray code masks for sequencing by hybridization,” *Genomics*, Vol. 23, 1994, pp. 233–235.
- [68] Alon, N., et al., “Equireplicate balanced binary codes for oligo arrays,” *SIAM J. Discrete Math.*, Vol. 14, No. 4, 2001, pp. 481–497.
- [69] Colbourn, C., A. Ling, and M. Tompa, “Construction of optimal quality control for oligo arrays,” *Bioinformatics*, Vol. 18, No. 4, 2002, pp. 529–535.
- [70] Dyachkov, A., and V. Rykov, “A survey of superimposed code theory,” *Problems in Control and Inform. Theory/Problemy Upravlen. Teor. Inform.*, Vol. 12, No. 4, 1983, pp. 229–242.
- [71] Milenkovic, O., “Joint quality- and error-control coding for DNA microarrays,” presented at the Inaugural Information Theory and Applications Workshop, San Diego, CA, Feb. 2006.

- [72] Gevertz, J., H. H. Gan, and T. Schlick, "In vitro RNA random pools are not structurally diverse: a computational analysis," *Bioinformatics*, Vol. 11, 2005, pp. 853–863.
- [73] Carothers, J. M., et al., "Informational complexity and functional activity of RNA structures," *J. Chem. Soc.*, Vol. 126, 2004, pp. 5130–5137.
- [74] Viennot, X. G., and M. Vauchassade de Chaumont, "Enumeration of RNA secondary structures by complexity," *Mathematics in Medicine and Biology, Lecture Notes in Biomaths.*, Vol. 57, 1985, pp. 360–365.
- [75] Waterman, M. S., "Combinatorics of RNA hairpins and cloverleaves," *Studies in Applied Math.*, Vol. 1, 1978, pp. 91–96.
- [76] Viennot, X. G., "A Strahler bijection between Dyck paths and planar trees," *Formal Power Series and Algebraic Combinatorics*, 1999, pp. 573–584.
- [77] Nebel, M., "Investigation of the Bernoulli model for RNA secondary structures," *Bull. Math. Biol.*, Vol. 66, No. 5, 2004, pp. 925–964.
- [78] Regnier, M., "Generating functions in computational biology," *INRIA Algorithms Seminar*, Mar. 1997.
- [79] Pergola, E., and R. Pinzani, "A combinatorial interpretation of the area of Schroeder paths," *Electron. J. Combin.*, Vol. 6, Research paper 40, 1999.
- [80] Immink, K. S., *Codes for Data Storage*, Englewood Cliffs, NJ: Prentice Hall, 1992.
- [81] Milenkovic, O., "Enumerating RNA motifs: a constrained coding approach," invited paper, 44th Allerton Conference on Communication, Control and Computing, Sept. 2006.
- [82] Gan, H., S. Pasquali, and T. Schlick, "Exploring the repertoire of RNA secondary motifs using graph theory: implications for RNA design," *Nucleic Acids Res.*, Vol. 31, 2003, pp. 2926–2943.
- [83] Fera, D., et al., "RAG: RNA-as-graphs web resource," *BMC Bioinformatics*, Vol. 5, 2004.

Application of Biomolecular Computing to Breakthroughs in Cryptography

Michael Shan-Hui Ho, Weng-Long Chang,
and Minyi Guo

13.1 Introduction

The problems of the NP-complete class are well known to be exponentially more difficult than evaluating determinants whose entries are merely numerical. When the problem size becomes large, it gets very difficult to solve these problems, even if very massive supercomputers are used. Some well-known examples are factoring, theorem-proving, and the traveling salesman problem.

The most ancient and basic problem of cryptography is to secure communication over an insecure channel. The traditional solution to this problem is called private key encryption. In private key encryption persons A and B hold a meeting before the remote transmission takes place and agree to use a pair of encryption and decryption algorithms E and D, and an additional piece of information S to be kept secret. However, the encryption system is no longer intended to be used only by a pair of pre-specified users, but by many senders wishing to send secret messages to a single recipient. The receiver B can publish authenticated information (called the public key) for anyone including the adversary, the sender A, and any other sender to read at their convenience. We call such an encryption method public key encryption.

The most basic primitive for cryptographic applications is a one-way function that is “easy” to compute but “hard” to invert. By “easy,” we mean that the function can be computed by a probabilistic polynomial time algorithm and by “hard” we mean that seems computationally infeasible. The RSA public-key encryption scheme is a candidate for a one-way trapdoor function. In 1977, Ron Rivest, Adi Shamir, and Len Adleman developed the public-key encryption scheme that is now known as RSA, after their initials. RSA [1] is the first incarnation of a public-key cryptosystem and is an algorithm that converts input data to an unrecognizable encryption, and converts the unrecognizable data back into its original decryption form. The construction of the RSA public-key cryptosystem is based on the ease of finding large prime numbers; security is based on the difficulty of factoring the product of two large prime numbers. The principal computation used for encryption and decryption is exponentiation in the RSA system.

The RSA public-key cryptosystem and two other papers [2, 3] are generally regarded as the seminal works in the field of public-key cryptography. The RSA system continues to occupy a central place in both the theoretical and practical development of the field. No method can be applied to break the RSA system in a reasonable amount of time. More than 400,000,000 copies of the RSA algorithm are currently installed, and it is the primary cryptosystem used for security on the Internet and the World Wide Web. Hence, RSA cryptography is a popular, highly secure algorithm for encrypting information using public and private keys.

Feynman first proposed molecular computation in 1961, but his idea was not implemented by experiment for a few decades [4]. In 1994 Adleman [5] succeeded in solving an instance of the Hamiltonian path problem in a test tube, just by handling DNA strands. Lipton [6] demonstrated that the Adleman techniques could be used to solve the satisfiability problem (the first NP-complete problem). Adleman and his co-authors [7] proposed a sticker-based model for decreasing the error rate of hybridization.

Through advances in molecular biology [8], it is now possible to produce roughly 10^{18} DNA strands that fit in a test tube. Those 10^{18} DNA strands can also be applied to represent 10^{18} bits of information. In the future if biological operations can deal with a tube of 10^{18} DNA strands and run without errors, then 10^{18} bits of information can simultaneously be correctly processed. Biological computing may then be able to provide a huge amount of parallelism for dealing with many computationally intensive problems in the real world.

The fastest supercomputers available today can execute approximately 10^{12} integer operations per second, which implies that (128×10^{12}) bits of information can be simultaneously processed in a second. The fastest supercomputers can process (128×10^{15}) bits of information in 1000 seconds. The extract operation is one of basic biological operations of the longest execution time. An extract operation can be done in approximately 1000 seconds. In the future if an extract operation can be used to deal with a tube with 10^{18} DNA strands and run without errors, then 10^{18} bits of information could simultaneously be correctly processed in 1000 seconds. At that time, basic biological operations may be faster than the fastest supercomputer. It has been pointed out [9] that storing information in molecules of DNA allows for an information density of approximately 1 bit per cu nm, a remarkable increase over traditional storage media such as videotape, which has an information density of approximately 1 bit per 10^{12} cu nm.

In this chapter, the RSA public-key cryptosystem is shown to be a breakthrough in using basic biological operations on a molecular computer. We demonstrate how to factor the product of two large prime numbers. First we construct solution spaces of good DNA strands to decrease the rate of errors for hybridization. By using basic biological operations, we develop three DNA-based algorithms to factor the product of two large prime numbers, parallel comparator, parallel subtractor, and parallel divider. After the product is factored, decoding an encrypted message is performed on a classical computer. Furthermore, this chapter indicates that public-key cryptosystems may not be secure and presents clear evidence of the ability of molecular computing to perform complicated mathematical operations.

The rest of this chapter is organized as follows: Section 13.2 introduces DNA models of computation proposed by Adleman and his co-authors and compares

them with other models. Section 13.3 introduces the DNA program to factor the product of two large prime numbers for solution spaces of DNA strands. Discussions and conclusions are then drawn in Section 13.4.

13.2 Introduction of DNA Background

In this section we review the basic structure of the DNA molecule and then discuss available techniques for dealing with DNA that will be used to break the RSA public-key cryptosystem. Simultaneously, several well-known DNA models are compared.

13.2.1 DNA Manipulations

DNA (*deoxyribonucleic acid*) is the *molecule* that plays the main role in DNA based computing [10]. Nucleotides are the structural units of DNA. In the most common nucleotides the base is a derivative of purine or pyrimidine, and five-carbon sugar. Purines include *adenine* and *guanine*, abbreviated A and G. Pyrimidines contain *cytosine* and *thymine*, abbreviated C and T. Because nucleotides are distinguished solely from their bases, they are simply represented as A, G, C, or T nucleotides, depending upon the kinds of bases they have.

In the past decade there have been revolutionary advances in the field of biomedical engineering, particularly in recombinant DNA and RNA manipulating. Due to the industrialization of the biotechnology field, laboratory techniques for recombinant DNA and RNA manipulation are becoming highly standardized. Basic principles about recombinant DNA can be found in [11–14]. In this subsection we describe eight biological operations that are useful for solving the problem of factoring integers. The method of constructing DNA solution space for the problem of factoring integers is based on the proposed method in [15, 16].

A (test) tube is a set of molecules of DNA (a multiset of finite strings over the alphabet {A, C, G, T}). Given a tube, one can perform the following biological operations:

1. *Extract*. Given a tube P and a short single strand of DNA, S , the operation produces two tubes $+(P, S)$ and $-(P, S)$, where $+(P, S)$ is all of the molecules of DNA in P that contain S as a substrand and $-(P, S)$ is all of the molecules of DNA in P that do not contain S .
2. *Merge*. Given tubes P_1 and P_2 , yield $\cup(P_1, P_2)$, where $\cup(P_1, P_2) = P_1 \cup P_2$. This operation is to pour two tubes into one, without any change in the individual strands.
3. *Detect*. Given a tube P , if P includes at least one DNA molecule we have “yes,” and if P contains no DNA molecule we have “no.”
4. *Discard*. Given a tube P , the operation will discard P .
5. *Amplify*. Given a tube P , the operation, $Amplify(P, P_1, P_2)$, will produce two new tubes P_1 and P_2 so that P_1 and P_2 are an exact copy of P (P_1 and P_2 are now identical) and P becomes an empty tube.
6. *Append*. Given a tube P containing a short strand of DNA, Z , the operation will append Z onto the end of every strand in P .

7. *Append-head*. Given a tube P containing a short strand of DNA, Z , the operation will append Z onto the head of every strand in P .
8. *Read*. Given a tube P , the operation is used to describe a single molecule, which is contained in tube P . Even if P contains many different molecules, each encoding a different set of bases, the operation can give an explicit description of exactly one of them.

13.2.2 Comparisons of Various Famous DNA Models

Based on solution space of *splint* in the Adleman-Lipton model, their methods [17–22] could be applied towards solving the traveling salesman problem, the dominating-set problem, the vertex cover problem, the clique problem, the independent-set problem, the three-dimensional matching problem, the set-packing problem, the set-cover problem, and the problem of exact cover by 3-sets. Lipton and his co-authors [23] indicated that DNA-based computing had been shown to easily be capable of breaking the data encryption standard from solution space of *splint*. The methods used for resolving problems have exponentially increased volumes of DNA and linearly increased the time.

Bach et al. [24] proposed a $n1.89^n$ volume, $O(n^2 + m^2)$ time molecular algorithm for the 3-coloring problem, and a 1.51^n volume, $O(n^2m^2)$ time molecular algorithm for the independent-set problem, where n and m are, subsequently, the number of vertices and the number of edges in the problems resolved. Fu [25] presented a polynomial-time algorithm with a 1.497^n volume for the 3-SAT problem, a polynomial-time algorithm with a 1.345^n volume for the 3-coloring problem, and a polynomial-time algorithm with a 1.229^n volume for the independent set. Although their size of those volumes is lower, constructing those volumes is more difficult and the time complexity is higher.

Quyang et al. [26] showed that enzymes could be used to solve the NP-complete clique problem. Because the maximum number of vertices that they can process is limited to 27, the maximum number of DNA strands for solving this problem is 2^{27} . Shin et al. [27] presented an encoding scheme for decreasing the error rate of hybridization. This method can be used in the traveling salesman problem to represent integers and real values with fixed-length codes. Arita et al. [28] and Morimoto et al. [29] proposed a new molecular experimental technique and a solid-phase method to find a Hamiltonian path. Amos [30] proposed a parallel filtering model for resolving the Hamiltonian path problem, the subgraph isomorphism problem, the 3-vertex-colorability problem, the clique problem, and the independent-set problem. Their proposed methods have lowered the error rate in real molecular experiments. In [31–33], the methods for DNA-based computing by self-assembly require the use of DNA nanostructures, called tiles, to produce expressive computational power and convenient input and output (I/O) mechanisms. That is, DNA tiles have lower error rate in self-assembly.

One of the earliest attempts to perform arithmetic operations (addition of two positive binary numbers) using DNA was by Guarneri et al. [34], utilizing the idea of encoding different bit values 0 and 1 as single-stranded DNAs, based upon their positions and the operands in which they appear. Vineet Gupta et al. [35] per-

formed logic and arithmetic operations using the fixed bit encoding of the full corresponding truth tables. Z. Frank Qiu and Mi Lu [36] applied substitution operation to insert results (by encoding all possible outputs of bit by bit operation along with second operand) in the operand strands. Ogihara and Ray [37], as well as Amos and Dunne [38], proposed methods to realize any Boolean circuit (with bounded fan) using DNA strands in a constructive fashion. Other new suggestions to perform all basic arithmetic operations are by Atanasiu [39] using P systems, by Frisco [40] using splicing operations under general H systems, and by Hug and Schuler [41]. Rana Barua et al. [42] proposed a recursive DNA algorithm for adding two binary numbers, which require $O(\log n)$ biosteps using only $O(n)$ different type of DNA strands, where n is the size of the binary string representing the larger of the two numbers.

A sticker-based model was proposed to reduce the error rate of hybridization in the Adleman-Lipton model. This model can be used for determining solutions of an instance in the set-cover problem. Simultaneously, Adleman and his co-authors [43] also pointed out that the data encryption standard could be easily broken from solution space of *stickers* in the sticker-based model. Perez-Jimenez et al. [44] employed the sticker-based model to resolve knapsack problems. In our previous work, Chang et al. [45–48] also employed the sticker-based model and the Adleman-Lipton model for dealing with Cook's theorem [49, 50], the set-splitting problem, the subset-sum problem, and the dominating-set problem for decreasing the error rate of hybridization.

13.3 Factoring the Product of Two Large Prime Numbers

13.3.1 Introduction to the RSA Public-Key Cryptosystem

In the RSA cryptosystem, a participant creates his public and secret keys with the following steps: 1. select two large random prime numbers p and q . 2. compute n by the equation $n = p * q$. 3. select a small odd integer e that is relatively prime to $\phi(n)$, which is equal to $(p - 1) * (q - 1)$. 4. compute d as the multiplicative inverse of e , module $\phi(n)$. 5. publish the pair $P = (e, n)$ as the RSA public key. 6. keep secret the pair $S = (d, n)$ as the secret key. A method to factor n as $p * q$ in a reasonable amount of time has not been found.

13.3.2 Solution Space of DNA Strands for Every Unsigned Integer

Suppose that an unsigned integer of k bits, M , is represented as a k -bit binary number, $m_k \dots m_1$, where the value of each bit m_j is either 1 or 0 for $1 \leq j \leq k$. The bits m_k and m_1 represent, respectively, the most significant bit and the least significant bit for M . The range of the value to an unsigned integer of k bits is from 0 to $2^k - 1$. In this chapter, each 15-base DNA sequence is used for every bit of the library. For every bit m_j , two *distinct* 15 base value sequences are designed. One represents the value “0” for m_j^1 and the other represents the value “1” for m_j^0 . The following algorithm is used to construct the solution space of DNA strands for 2^k different unsigned integer values.

```

Procedure InitialSolution( $T_0$ )
  (1) For  $j = k$  down to 1
    (1a) Amplify( $T_0, T_1, T_2$ ).
    (1b) Append( $T_1, m_j^1$ ).
    (1c) Append( $T_2, m_j^0$ ).
    (1d)  $T_0 = \cup(T_1, T_2)$ .
  EndFor
EndProcedure

```

The algorithm, InitialSolution(T_0), is implemented by means of the *amplify*, *append*, and *merge* operations. Each execution of Step (1a) is used to amplify tube T_0 and to generate two new tubes, T_1 and T_2 , which are copies of T_0 . Tube T_0 then becomes empty. Then, Step (1b) is applied to append a DNA sequence, representing the value “1” for m_j , onto the end of every strand in tube T_1 . This is to say that those integers containing the value “1” to the j^{th} bit appear in tube T_1 . Step (1c) is also employed to append a DNA sequence, representing the value “0” for m_j , onto the end of every strand in tube T_2 . That implies that these integers containing the value “0” to the j^{th} bit appear in tube T_2 . Next, Step (1d) is used to pour tubes T_1 and T_2 into tube T_0 . This indicates that DNA strands in tube T_0 include DNA sequences of $m_j = 1$ and $m_j = 0$. At the end of Steps (1a–1d), tube T_0 consists of 2^k DNA sequences, representing 2^k different unsigned integer values.

Consider that the number of bits for M is three bits. Eight values for M are then 000, 001, 010, 011, 100, 101, 110, and 111. Tube T_0 is an empty tube and is regarded as an input tube for the algorithm, InitialSolution(T_0). Because the value for k is 3, Steps (1a–1d) will be run three times. After the first execution of Step (1a), tube $T_0 = \phi$, tube $T_1 = \phi$, and tube $T_2 = \phi$. Next, after the first execution of Steps (1b and 1c), tube $T_1 = \{m_3^1\}$ and tube $T_2 = \{m_3^0\}$. After the first execution of Step (1d), tube $T_0 = \{m_3^1, m_3^0\}$, tube $T_1 = \phi$, and tube $T_2 = \phi$. Then, after the second execution of Step (1a), tube $T_0 = \phi$, tube $T_1 = \{m_3^1, m_3^0\}$, and tube $T_2 = \{m_3^1, m_3^0\}$. After the rest of operations are performed, tube $T_1 = \phi$, tube $T_2 = \phi$, and the result for tube T_0 is shown in Table 13.1.

13.3.3 Construction of the Product for Two Large Prime Numbers

Assume that the length for n , the product of two large prime numbers of k bits, denoted in Subsection 13.3.1, is $(2 * k)$ bits. Also suppose that the product, n , is used to represent the minuend (dividend) and the difference for successive compare, shift, and subtract operations in a divider. When n is divided by M , an unsigned integer of k bits denoted in Subsection 13.3.2, M is one of two large prime numbers if the remainder is equal to zero. Assume that in a divider the length of a dividend is $(2 * k)$ bits and the length of a divisor is d bits, where $1 \leq d \leq k$. It is very obvi-

Table 13.1 The result for tube T_0 is generated by the algorithm, InitialSolution(T_0).

Tube	Result generated by InitialSolution(T_0)
T_0	$\{m_3^1 m_2^1 m_1^1, m_3^1 m_2^1 m_1^0, m_3^1 m_2^0 m_1^1, m_3^1 m_2^0 m_1^0, m_3^0 m_2^1 m_1^1, m_3^0 m_2^1 m_1^0, m_3^0 m_2^0 m_1^1, m_3^0 m_2^0 m_1^0\}$

ous that the division instruction is finished through successive compare, shift, and subtract operations of at most $(2 * k)$ times. Therefore, suppose that n is represented as a $(2 * k)$ -bit binary number, $n_{o, (2 * k)} \dots n_{o, 1}$, where the value of each bit $n_{o, q}$ is either 1 or 0 for $1 \leq o \leq (2 * k + 1)$ and $1 \leq q \leq (2 * k)$. The bits, $n_{o, (2 * k)}$ and $n_{o, 1}$ respectively represent the most significant bit and the least significant bit for n . Two binary numbers, $n_{o, (2 * k)} \dots n_{o, 1}$ and $n_{o+1, (2 * k)} \dots n_{o+1, 1}$ are respectively applied to represent the minuend and the difference for the successive compare, shift, and subtract operations of the o^{th} time. That is, the binary number $n_{o+1, (2 * k)} \dots n_{o+1, 1}$ is the minuend for the successive compare, shift, and subtract operations of the $(o + 1)^{th}$ time.

For every bit $n_{o, q}$, two distinct 15-base-value sequences are designed. One represents the value “0” for $n_{o, q}$ and the other represents the value “1” for $n_{o, q}$. The following algorithm is used to construct a DNA strand for the value of n .

```

Procedure InitialProduct( $T_0$ )
  (1) For  $q = 1$  to  $2 * k$ 
    (1a) Append-head( $T_0, n_{1, q}$ ).
  EndFor
EndProcedure

```

From InitialProduct(T_0), it takes $(2 * k)$ *append-head* operations and one test tube to construct a DNA strand. Consider that the number of bits for n is six bits and the value for n is 001111. Tube T_0 , with the result shown in Table 13.1, is regarded as an input tube for the algorithm, InitialProduct(T_0). Because the value for $2 * k$ is six, Step (1a) will be executed six times. After each operation for Step (1a), the result is shown in Table 13.2.

13.3.4 Construction of a Parallel Comparator

A division operation for a dividend of $(2 * k)$ bits and a divisor of d bits for $1 \leq d \leq k$ are carried out by successive compare, shift, and subtract operations at most $(2 * k + 1)$ times. This indicates that compare and shift operations must be finished before the corresponding subtraction operation is done. Therefore, the algorithm, OneBitComparator($T_0^>, T_0^=, T_0^<, d, o, j$), is presented to perform the function of a one-bit parallel comparator and the algorithm, ParallelComparator($T_0, T_0^>, T_0^=, T_0^<, d, o$), is proposed to perform the function of a k -bit parallel comparator.

```

Procedure OneBitComparator( $T_0^>, T_0^=, T_0^<, d, o, j$ )
  (1)  $T_1 = +(T_0^=, n_{o, (2 * k) - (o - 1) - (j - o)^1})$  and  $T_2 = -(T_0^=, n_{o, (2 * k) - (o - 1) - (j - o)^1})$ .
  (2)  $T_3 = +(T_1, m_{(k - d + 1) + o - j^1})$  and  $T_4 = -(T_1, m_{(k - d + 1) + o - j^1})$ .
  (3)  $T_5 = +(T_2, m_{(k - d + 1) + o - j^1})$  and  $T_6 = -(T_2, m_{(k - d + 1) + o - j^1})$ .
  (4)  $T_0^= = \cup(T_0^=, T_3, T_6)$ .
  (5)  $T_0^> = \cup(T_0^>, T_4)$ .
  (6)  $T_0^< = \cup(T_0^<, T_5)$ .
EndProcedure

```


Table 13.2 The result for tube T_0 is generated by the algorithm, InitialProduct(T_0).

Tube	Result generated by InitialProduct(T_0)
T_0	$\{..., n_{1,6}^0 n_{1,5}^0 n_{1,4}^1 n_{1,3}^1 n_{1,2}^1 n_{1,1}^1 m_3^1 m_2^0 m_1^1, ...\}$

The algorithm, OneBitComparator($T_0^>, T_0^=, T_0^<, d, o, j$), is implemented by the *extract* and *merge* operations. The execution of Step (1) employs the *extract* operation to form two test tubes: T_1 and T_2 . The first tube T_1 includes all of the strands that have $n_{o, (2 * k) - (o - 1) - (j - o)} = 1$. The second tube T_2 consists of all of the strands that have $n_{o, (2 * k) - (o - 1) - (j - o)} = 0$. Next, on the execution of Step (2), it also uses the *extract* operation to form two test tubes: T_3 and T_4 . The first tube T_3 includes all of the strands that have $n_{o, (2 * k) - (o - 1) - (j - o)} = 1$ and $m_{(k - d + 1) + o - j} = 1$. The second tube T_4 consists of all of the strands that have $n_{o, (2 * k) - (o - 1) - (j - o)} = 1$ and $m_{(k - d + 1) + o - j} = 0$. The execution of Step (3) uses the *extract* operation to form two test tubes: T_5 and T_6 . The first tube T_5 includes all of the strands that have $n_{o, (2 * k) - (o - 1) - (j - o)} = 0$ and $m_{(k - d + 1) + o - j} = 1$. The second tube T_6 consists of all of the strands that have $n_{o, (2 * k) - (o - 1) - (j - o)} = 0$ and $m_{(k - d + 1) + o - j} = 0$. Because the corresponding bits of the dividend and the divisor in T_3 are both one, and the corresponding bits of the dividend and the divisor in T_6 are both zero; next, the execution of Step (4) uses the *merge* operations to pour T_3 and T_6 into $T_0^=$. In T_4 , the corresponding bit of the dividend is one and the corresponding bit of the divisor is zero, so the execution of Step (5) also applies the *merge* operations to pour T_4 into $T_0^>$. Next, in T_5 , since the corresponding bit of the dividend is zero and the corresponding bit of the divisor is one, the execution of Step (6) employs the *merge* operations to pour T_5 into $T_0^<$. From OneBitComparator($T_0^>, T_0^=, T_0^<, d, o, j$), it takes three *extract* operations, three *merge* operations, and nine test tubes to finish the function of a one-bit parallel comparator.

Procedure ParallelComparator($T_0, T_0^>, T_0^=, T_0^<, d, o$)

```

(1) For  $j = 1$  to  $o - 1$ 
    (1a)  $T_7 = + (T_0, n_{o, (2 * k) - (j - 1)^1})$  and  $T_8 = -(T_0, n_{o, (2 * k) - (j - 1)^1})$ .
    (1b)  $T_0^> = \cup(T_0^>, T_7)$ .
    (1c) If (Detect( $T_8$ ) = "yes") then
    (1d)  $T_0 = \cup(T_0, T_8)$ .
        Else
    (1e) Terminate the algorithm.
    EndIf
EndFor
(2)  $T_0^= = \cup(T_0^=, T_0)$ .
(3) For  $j = o$  to  $k + o - d$ 
    (3a) OneBitComparator( $T_0^>, T_0^=, T_0^<, d, o, j$ ).
    (3b) If (Detect( $T_0^=$ ) = "no") then
    (3c) Terminate the algorithm.
    EndIf
EndFor
EndProcedure

```


Step (1) is the first loop and is used to compare the most significant $(o - 1)$ bits of the dividend with $(o - 1)$ zeros for the o^{th} compare and shift operations. The first execution of Step (1a) employs the *extract* operation to form two test tubes: T_7 and T_8 . The first tube T_7 includes all of the strands that have $n_{o, (2 * k) - (j - 1)} = 1$. The second tube T_8 consists of all of the strands that have $n_{o, (2 * k) - (j - 1)} = 0$. In T_7 , the corresponding bit of the dividend is one and the shift bit of the divisor is zero, so the first execution of Step (1b) uses the *merge* operations to pour T_7 into $T_0^>$. The first execution of Step (1c) employs the *detect* operations to check whether tube T_8 contains any DNA strand or not. If a “yes” is returned, then the first execution of Step (1d) applies the *merge* operations to pour T_8 into T_0 . Otherwise, the algorithm is terminated in Step (1e). Repeat the execution of each step in the loop until the number of the execution for the loop is performed.

After each operation in the first loop is finished, tube T_0 contains the strands that have the comparative result (“=”) for the most significant $(o - 1)$ bits of the dividend with $(o - 1)$ zeros for the o^{th} compare, and shift operations. Step (2) uses the *merge* operation to pour T_0 into $T_0^=$. The first execution of Step (3a) calls the algorithm, $\text{OneBitComparator}(T_0^>, T_0^=, T_0^<, d, o, j)$, to finish the comparative result of the corresponding bit for the $(2 * k)$ -bit dividend and the d -bit divisor for $1 \leq d \leq k$ in a divider. After Step (3a) is performed, the comparative results are respectively represented in $T_0^>$, $T_0^=$, and $T_0^<$. On the first execution of Step (3b), it uses the *detect* operations to check whether there is any DNA sequence in $T_0^=$. If a “no” is returned, then the execution of Step (3c) is used to terminate the algorithm. Otherwise, Steps (3a) and (3b) are repeated until the corresponding bits of the $(2 * k)$ -bit dividend and the d -bit divisor for $1 \leq d \leq k$ in a divider are all processed. Finally, tube $T_0^>$ contains the strands with the comparative result of greater than (“>”), tube $T_0^=$ includes the strands with the comparative result of equal (“=”), and tube $T_0^<$ consists of the strands with the comparative result of less than (“<”). From $\text{ParallelComparator}(T_0, T_0^>, T_0^=, T_0^<, d, o)$, it takes $(3 * k - 3 * d + o + 2)$ *extract* operations, $(3 * k - 3 * d + 2 * o + 2)$ *merge* operations, $(k - d + o)$ *detect* operations, and eleven tubes to finish the function of a k -bit parallel comparator.

13.3.5 Construction of a Parallel One-Bit Subtractor

A one-bit subtractor is a function that forms the arithmetic subtraction of three input bits. It consists of three inputs and two outputs. Two of the input bits respectively represent minuend and subtrahend bits to be subtracted. The third input represents the borrow bit from the previous higher significant position. The first output gives the value of the difference for minuend and subtrahend bits to be subtracted. The second output gives the value of the borrow bit to minuend and subtrahend bits to be subtracted. The truth table of the one-bit subtractor is as follows (Table 13.3).

Suppose that a one-bit binary number $n_{o, q}$ denoted in Subsection 13.3.3 is used to represent the first input of a one-bit subtractor for $1 \leq o \leq (2 * k + 1)$ and $1 \leq q \leq (2 * k)$. Also assume that a one-bit binary number $n_{o + 1, q}$ denoted in Subsection 13.3.3 is applied to represent the first output of a one-bit subtractor. Suppose that a one-bit binary number m_j denoted in Subsection 13.3.2 is also employed to represent the second input of a one-bit subtractor for $1 \leq j \leq k$. Also assume that a one-bit binary number $b_{o, q}$ is employed to represent the second output of a one-bit

Table 13.3 The truth table of a one-bit subtractor.

Minuend bit	Subtrahend bit	Previous borrow bit	Difference bit	Borrow bit
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

subtractor. Also suppose that a one-bit binary number $b_{o,q-1}$ is employed to represent the third input of a one-bit subtractor.

For every bit $b_{o,q-1}$ and $b_{o,q}$ to $1 \leq o \leq (2 * k + 1)$ and $1 \leq q \leq (2 * k)$, two *distinct* DNA sequences are designed to represent the value “0” or “1” of every corresponding bit. For convenience, we assume that $b_{o,q}^1$ defines the value of 1 and $b_{o,q}^0$ defines the value of 0. Also suppose that $n_{o+1,q}^1$ defines the value of 1 and $n_{o+1,q}^0$ defines the value of 0. Similarly, assume that $b_{o,q-1}^1$ defines the value of 1 and $b_{o,q-1}^0$ defines the value of 0. The following algorithm is proposed to finish the function of a parallel one-bit subtractor.

Procedure ParallelOneBitSubtractor($T_0^{>=}$, o , q , j)

- (1) $T_1 = +(T_0^{>=}, n_{o,q}^1)$ and $T_2 = -(T_0^{>=}, n_{o,q}^1)$.
- (2) $T_3 = +(T_1, m_j^1)$ and $T_4 = -(T_1, m_j^1)$.
- (3) $T_5 = +(T_2, m_j^1)$ and $T_6 = -(T_2, m_j^1)$.
- (4) $T_7 = +(T_3, b_{o,q-1}^1)$ and $T_8 = -(T_3, b_{o,q-1}^1)$.
- (5) $T_9 = +(T_4, b_{o,q-1}^1)$ and $T_{10} = -(T_4, b_{o,q-1}^1)$.
- (6) $T_{11} = +(T_5, b_{o,q-1}^1)$ and $T_{12} = -(T_5, b_{o,q-1}^1)$.
- (7) $T_{13} = +(T_6, b_{o,q-1}^1)$ and $T_{14} = -(T_6, b_{o,q-1}^1)$.
 - (8a) If (Detect(T_7) = “yes”) then
- (8) Append-head($T_7, n_{o+1,q}^1$) and Append-head($T_7, b_{o,q}^1$).
 - EndIf
 - (9a) If (Detect(T_8) = “yes”) then
- (9) Append-head($T_8, n_{o+1,q}^0$) and Append-head($T_8, b_{o,q}^0$).
 - EndIf
 - (10a) If (Detect(T_9) = “yes”) then
- (10) Append-head($T_9, n_{o+1,q}^0$) and Append-head($T_9, b_{o,q}^0$).
 - EndIf
 - (11a) If (Detect(T_{10}) = “yes”) then
- (11) Append-head($T_{10}, n_{o+1,q}^1$) and Append-head($T_{10}, b_{o,q}^0$).
 - EndIf
 - (12a) If (Detect(T_{11}) = “yes”) then
- (12) Append-head($T_{11}, n_{o+1,q}^0$) and Append-head($T_{11}, b_{o,q}^1$).
 - EndIf
 - (13a) If (Detect(T_{12}) = “yes”) then
- (13) Append-head($T_{12}, n_{o+1,q}^1$) and Append-head($T_{12}, b_{o,q}^1$).
 - EndIf

```

    (14a) If (Detect( $T_{13}$ ) = "yes") then
    (14) Append-head( $T_{13}$ ,  $n_o + 1, q^1$ ) and Append-head( $T_{13}$ ,  $b_o, q^1$ ).
    (15a) If (Detect( $T_{14}$ ) = "yes") then
    (15) Append-head( $T_{14}$ ,  $n_o + 1, q^0$ ) and Append-head( $T_{14}$ ,  $b_o, q^0$ ).
    EndIf
    (16)  $T_0^{>=}$  =  $\cup(T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{13}, T_{14})$ .
EndProcedure

```

The algorithm, $\text{ParallelOneBitSubtractor}(T_0^{>=}, o, q, j)$, is implemented by means of the *extract*, *append-head*, and *merge* operations. The execution of Step (1) employs the *extract* operation to form two test tubes: T_1 and T_2 . The first tube T_1 includes all of the strands that have $n_{o,q} = 1$. The second tube T_2 consists of all of the strands that have $n_{o,q} = 0$. In Step (2), the *extract* operation is used to form two test tubes: T_3 and T_4 . The first tube T_3 includes all of the strands that have $n_{o,q} = 1$ and $m_j = 1$. The second tube T_4 consists of all of the strands that have $n_{o,q} = 1$ and $m_j = 0$. Next, the execution of Step (3) uses the *extract* operation to form two test tubes: T_5 and T_6 . The first tube T_5 includes all of the strands that have $n_{o,q} = 0$ and $m_j = 1$. The second tube T_6 consists of all of the strands that have $n_{o,q} = 0$ and $m_j = 0$. The execution of Step (4) uses the *extract* operation to form two test tubes: T_7 and T_8 . The first tube T_7 includes all of the strands that have $n_{o,q} = 1$, $m_j = 1$, and $b_{o,q-1} = 1$. The second tube T_8 consists of all of the strands that have $n_{o,q} = 1$, $m_j = 1$, and $b_{o,q-1} = 0$. Then, on the execution of Step (5), it applies the *extract* operation to form two test tubes: T_9 and T_{10} . The first tube T_9 includes all of the strands that have $n_{o,q} = 1$, $m_j = 0$, and $b_{o,q-1} = 1$. The second tube T_{10} consists of all of the strands that have $n_{o,q} = 1$, $m_j = 0$, and $b_{o,q-1} = 0$. On the execution of Step (6), it employs the *extract* operation to form two test tubes: T_{11} and T_{12} . The first tube T_{11} includes all of the strands that have $n_{o,q} = 0$, $m_j = 1$, and $b_{o,q-1} = 1$. The second tube T_{12} consists of all of the strands that have $n_{o,q} = 0$, $m_j = 1$, and $b_{o,q-1} = 0$. Next, the execution of Step (7) uses the *extract* operation to form two test tubes: T_{13} and T_{14} . The first tube T_{13} includes all of the strands that have $n_{o,q} = 0$, $m_j = 0$, and $b_{o,q-1} = 1$. The second tube T_{14} consists of all of the strands that have $n_{o,q} = 0$, $m_j = 0$, and $b_{o,q-1} = 0$. After finishing Steps (1–7), eight different inputs of a one-bit subtractor in Table 13.3 have been poured into tubes T_7 through T_{14} respectively.

Steps (8a–15a) are used to check whether contains any DNA strand for tubes $T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{13}$, and T_{14} respectively. If a “yes” is returned for any of those steps, then the corresponding *append-head* operations will be run. Next, the execution of Step (8) uses the *append-head* operations to append $n_o + 1, q^1$ and b_o, q^1 onto the head of every strand in T_7 . On the execution of Step (9), it applies the *append-head* operations to append $n_o + 1, q^0$ and b_o, q^0 onto the head of every strand in T_8 . Then, the execution of Step (10) employs the *append-head* operations to append $n_o + 1, q^0$ and b_o, q^0 onto the head of every strand in T_9 . On the execution of Step (11), it uses the *append-head* operations to append $n_o + 1, q^1$ and b_o, q^1 onto the head of every strand in T_{10} . Next, the execution of Step (12) uses the *append-head* operations to append $n_o + 1, q^0$ and b_o, q^1 onto the head of every strand in T_{11} . On the execution of Step (13), it uses the *append-head* operations to append $n_o + 1, q^1$ and b_o, q^1 onto the head of every strand in T_{12} . Then, the execution of Step (14) applies the *append-head* operations to append $n_o + 1, q^1$ and b_o, q^1 onto the head of every strand in T_{13} . On the execution of Step (15), it employs the *append-head* operations to ap-

pend $n_{o+1,q}^0$ and $b_{o,q}^0$ onto the head of every strand in T_{14} . After finishing Steps (8–15), eight different outputs of a one-bit subtractor in Table 13.3 are appended into tubes T_7 through T_{14} respectively. Finally, the execution of Step (16) applies the *merge* operation to pour tubes T_7 through T_{14} into $T_0^{>=}$. Tube $T_0^{>=}$ contains the strands finishing the subtraction operations of a bit.

From $\text{ParallelOneBitSubtractor}(T_0^{>=}, o, q, j)$, it takes seven *extract* operations, sixteen *append-head* operations, sixteen *detect* operations, one *merge* operation, and fifteen test tubes to compute the subtraction of a bit. Two output bits of a one-bit subtractor encode the difference bit and the borrow bit to the subtraction of a bit.

13.3.6 Construction of a Binary Parallel Subtractor

The one-bit subtractor introduced in Subsection 13.3.5 determines the difference bit and the borrow bit for two input bits and a previous borrow bit. A minuend of k bits and a subtrahend of d bits for $1 \leq d \leq k$ can finish subtractions of at most k times by means of this one-bit subtractor. A binary parallel subtractor is a function that performs the arithmetic subtraction for a minuend of k bits and a subtrahend of d bits for $1 \leq d \leq k$. The following algorithm is proposed to finish the function of a binary parallel subtractor.

Procedure $\text{BinaryParallelSubtractor}(T_0^{>=}, d, o, q)$

- (1) For $j = 1$ to $k - d + 1$
 - (1a) $\text{ParallelOneBitSubtractor}(T_0^{>=}, o, 2 * k - (o - 1) - (k - d + 1 - j), j)$.
- EndFor
- (2) For $q = (2 * k) - (o - 1) + 1$ to $2 * k$
 - (2a) $T_{21} = +(T_0^{>=}, n_{o,q}^1)$ and $T_{22} = -(T_0^{>=}, n_{o,q}^1)$.
 - (2b) $T_{23} = +(T_{21}, b_{o,q-1}^1)$ and $T_{24} = -(T_{21}, b_{o,q-1}^1)$.
 - (2c) $T_{25} = +(T_{22}, b_{o,q-1}^1)$ and $T_{26} = -(T_{22}, b_{o,q-1}^1)$.
 - (2d) If $(\text{Detect}(T_{23}) = \text{"yes"})$ then
 - (2d0) $\text{Append-head}(T_{23}, n_{o+1,q}^0)$ and $\text{Append-head}(T_{23}, b_{o,q}^0)$.
 - EndIf
 - (2e) If $(\text{Detect}(T_{24}) = \text{"yes"})$ then
 - (2e0) $\text{Append-head}(T_{24}, n_{o+1,q}^1)$ and $\text{Append-head}(T_{24}, b_{o,q}^0)$.
 - EndIf
 - (2f) If $(\text{Detect}(T_{25}) = \text{"yes"})$ then
 - (2f0) $\text{Append-head}(T_{25}, n_{o+1,q}^1)$ and $\text{Append-head}(T_{25}, b_{o,q}^1)$.
 - EndIf
 - (2g) If $(\text{Detect}(T_{26}) = \text{"yes"})$ then
 - (2g0) $\text{Append-head}(T_{26}, n_{o+1,q}^0)$ and $\text{Append-head}(T_{26}, b_{o,q}^0)$.
 - EndIf
 - (2h) $T_0^{>=} = \cup(T_{23}, T_{24}, T_{25}, T_{26})$.
- EndFor

EndProcedure

Step (1) is the only loop and is used mainly to finish the function of a binary parallel subtractor. On the first execution of Step (1a), it calls the procedure, Par-

allelOneBitSubtractor $[T_0^{>=}, o, 2 * k - (o - 1) - (k - d + 1 - j), j]$, to compute the arithmetic subtraction of the least significant bit to the minuend and the subtrahend with the result left in $T_0^{>=}$. Step (1a) is repeated until the most significant bit in the minuend and the subtrahend is processed. Tube $T_0^{>=}$ contains the strands finishing the subtraction operations of at most k bits.

Because after each operation in Step (1) is performed, the borrow bit, $b_{o, (2 * k) - (o - 1)}$, is perhaps one or zero. If its value is one, then this implies that the corresponding dividend in every bit pattern in tube $T_0^{>=}$ should be subtracted by one. Therefore, Step (2) is the second main loop and is used to perform the operation of decrease. On each execution from Step (2a) through (2c), the *extract* operations are applied to form some different tubes. This is to say that tube T_{23} includes all of the strands that have $n_{o, q} = 1$ and $b_{o, q - 1} = 1$, tube T_{24} consists of all of the strands that have $n_{o, q} = 1$ and $b_{o, q - 1} = 0$, tube T_{25} includes all of the strands that have $n_{o, q} = 0$ and $b_{o, q - 1} = 1$, tube T_{26} consists of all of the strands that have $n_{o, q} = 0$ and $b_{o, q - 1} = 0$, tube $T_{21} = \emptyset$, and tube $T_{22} = \emptyset$. After finishing Steps (2a–2c), four different inputs of a one-bit subtractor in Table 13.3 have been poured into tubes T_{23} through T_{26} respectively.

Each execution for Steps (2d–2g) is used to check whether it contains any DNA strand for tubes T_{23} , T_{24} , T_{25} , and T_{26} respectively. If any a “yes” is returned from those steps, then the corresponding *append-head* operations for Steps (2d0), (2e0), (2f0), and (2g0) will be run and those values, $n_{o + 1, q}^1$, $n_{o + 1, q}^0$, $b_{o, q}^0$, and $b_{o, q}^1$, are appended onto the head of every strand in the corresponding tubes. After finishing those steps, four different outputs of a one-bit subtractor in Table 13.3 are appended into tubes T_{23} through T_{26} respectively. Next, each execution for Step (2h) applies the *merge* operation to pour tubes T_{23} through T_{26} into $T_0^{>=}$. At the end of Step (2), tube $T_0^{>=}$ contains the strands finishing the subtraction operations of $(2 * k)$ bits.

13.3.7 Construction of a Binary Parallel Divider

A binary parallel divider is a function that performs the arithmetic division for a dividend of $(2 * k)$ bits and a divisor of d bits for $1 \leq d \leq k$. The quotient obtained from the dividend and the divisor can be at most up to $(2 * k)$ bits long. The remainder obtained from the dividend and the divisor can also be at most up to k bits long. Because we only check whether the remainder is equal to zero, therefore, the quotient can be ignored. The following algorithm is proposed to finish the function of a binary parallel divider. The second parameter, d , in the procedure is used to represent the d^{th} division operation.

Procedure BinaryParallelDivider(T_0 , d)

- (1) For $o = 1$ to $k + d$
 - (1a0) Append-head(T_0 , b_o , 0^0).
 - (1a) ParallelComparator(T_0 , $T_0^{>}$, $T_0^=$, $T_0^{<}$, d , o).
 - (1b) $T_0^{>=} = \cup(T_0^{>}, T_0^=)$.
 - (1c) If (Detect($T_0^{>=}$) = “yes”) then
- (2) For $q = 1$ to $(2 * k) - (o - 1) - (k - d) - 1$
 - (2a) $T_1 = +(T_0^{>=}, n_{o, q}^1)$ and $T_2 = -(T_0^{>=}, n_{o, q}^1)$.

```

(2a1) If (Detect( $T_1$ ) = "yes") then
  (2b) Append-head( $T_1, n_o + 1, q^1$ ) and Append-head( $T_1, b_o, q^0$ ).
  EndIf
  (2b1) If (Detect( $T_2$ ) = "yes") then
    (2c) Append-head( $T_2, n_o + 1, q^0$ ) and Append-head( $T_2, b_o, q^0$ ).
    EndIf
    (2d)  $T_0^{>=}$  =  $\cup(T_1, T_2)$ .
  EndFor
(3) BinaryParallelSubtractor( $T_0^{>=}$ ,  $d, o, q$ ).
  EndIf
(4) If (Detect( $T_0^{<}$ ) = "yes") then
(5) For  $q = 1$  to  $2 * k$ 
  (5a)  $T_1 = +(T_0^{<}, n_o, q^1)$  and  $T_2 = -(T_0^{<}, n_o, q^1)$ .
  (5a1) If (Detect( $T_1$ ) = "yes") then
    (5b) Append-head( $T_1, n_o + 1, q^1$ ) and Append-head( $T_1, b_o, q^0$ ).
    EndIf
    (5b1) If (Detect( $T_2$ ) = "yes") then
      (5c) Append-head( $T_2, n_o + 1, q^0$ ) and Append-head( $T_2, b_o, q^0$ ).
      EndIf
      (5d)  $T_0^{<}$  =  $\cup(T_1, T_2)$ .
    EndFor
  EndIf
(6)  $T_0 = \cup(T_0^{>=}, T_0^{<})$ .
  EndFor
EndProcedure

```

The division to a dividend of $(2 * k)$ bits and a divisor of d bits for $1 \leq d \leq k$ is finished through successive compare, shift, and subtract operations of at most $(2 * k)$ times. In the first compare, shift, and subtract operations, the least significant position for the dividend and the divisor is subtracted, and the input borrow bit must be 0. Step (1) is the main loop and is applied to finish the function of a binary parallel divider. So, each execution of Step (1a0) uses the *append-head* operation to append 15-based DNA sequences representing $b_o, 0^0$ onto the head of every strand in T_0 . On each execution of Step (1a), it calls *ParallelComparator*($T_0, T_0^{>}, T_0^{=}, T_0^{<}, d, o$) to compare the divisor with the corresponding bits of the dividend. After it is finished, three tubes are generated: $T_0^{>}$, $T_0^{=}$, and $T_0^{<}$. The first tube $T_0^{>}$ includes the strands with the comparative result of greater than (" $>$ "). The second tube $T_0^{=}$ includes the strands with the comparative result of equal (" $=$ "). The third tube $T_0^{<}$ consists of the strands with the comparative result of less than (" $<$ "). Next, each execution of Step (1b) employs the *merge* operation to pour tubes $T_0^{>}$ and $T_0^{=}$ into $T_0^{>=}$. On each execution Step (1c) applies the *detect* operation to check whether tube $T_0^{>=}$ contains any DNA strand. If a "yes" is returned, then Step (2) through Step (4a) will be run. Otherwise, those steps will not be executed. Step (2) is a loop and is used mainly to reserve the least significant $[(2 * k) - (o - 1) - (k - d) - 1]$ bits of the dividend. This implies that the least significant $[(2 * k) - (o - 1) - (k - d) - 1]$ bits of the minuend (dividend) for the o^{th} compare, shift, and subtract operations are reserved. And they are equal to the least significant $[(2 * k) - (o - 1) - (k - d) -$

1] bits of the difference for the same operations. Therefore, on each execution of Step (2a), it uses the *extract* operation to form two test tubes: T_1 and T_2 . The first tube T_1 includes all of the strands that have $n_{o,q} = 1$. The second tube T_2 consists of all of the strands that have $n_{o,q} = 0$. On each execution Step (2a1) uses the *detect* operation to test if tube T_1 contains any DNA strand. If a “yes” is returned, then Step (2b) will be run. Otherwise, that step will not be executed. Next, each execution of Step (2b) uses the *append-head* operations to append $n_{o+1,q}^1$ and $b_{o,q}^0$ onto the head of every strand in T_1 . Each execution of Step (2b1) applies the *detect* operation to examine if tube T_2 contains any DNA strand. If a “yes” is returned, then Step (2c) will be run. Otherwise, that step will not be executed. On each execution of Step (2c), it applies the *append-head* operations to append $n_{o+1,q}^0$ and $b_{o,q}^0$ onto the head of every strand in T_2 . Then, each execution of Step (2d) employs the *merge* operation to pour tubes T_1 and T_2 into $T_0^{>=}$. Tube $T_0^{>=}$ contains the strands finishing compare, shift, and subtract operations of a bit. Repeat execution of Steps (2a–2d) until the least significant $[(2 * k) - (o - 1) - (k - d) - 1]$ bits of the minuend (dividend) are processed. Tube $T_0^{>=}$ contains the strands finishing compare, shift, and subtract operations of the least significant $[(2 * k) - (o - 1) - (k - d) - 1]$ bits of the minuend (dividend).

Next, each execution of Step (3) calls the algorithm, *BinaryParallelSubtractor*($T_0^{>=}, d, o, q$), to finish compare, shift, and subtract operations of $(k - d + 1)$ bits. Step (4) is a loop and it is used to finish compare, shift, and subtract operations of the most significant $(o - 1)$ bits in the minuend (dividend). Because the most significant $(o - 1)$ bits in the minuend (dividend) for the o^{th} compare, shift, and subtract operations are all zero, the most significant $(o - 1)$ bits of the difference to the o^{th} compare, shift, and subtract operations are equal to the most significant $(o - 1)$ bits of the minuend to the same operations. On each execution of Step (4a), it applies the *append-head* operations to append $n_{o+1,q}^0$ and $b_{o,q}^0$ onto the head of every strand in $T_0^{>=}$. Repeat execution of Step (4a) until the most significant $(o - 1)$ bits of the minuend are processed. Tube $T_0^{>=}$ contains the strands finishing the o^{th} compare, shift, and subtract operations for the comparative result of greater than or equal to (“>=”).

Next, each execution of Step (4b) applies the *detect* operation to check whether tube $T_0^{<}$ contains any DNA strand. If a “yes” is returned, then Step (5–5d) will be run. Otherwise, those steps will not be executed. Hence, $T_0^{<}$ consists of all of the strands with the comparative result of less than (“<”). This implies that the $(2 * k)$ bits of the difference to the o^{th} compare, shift, and subtract operations are equal to the $(2 * k)$ bits of the minuend to the same operations. Step (5) is a loop and is employed to finish the o^{th} compare, shift, and subtract operations for tube $T_0^{<}$. On each execution of Step (5a), it employs the *extract* operation to form two test tubes: T_1 and T_2 . The first tube T_1 includes all of the strands that have $n_{o,q} = 1$. The second tube T_2 consists of all of the strands that have $n_{o,q} = 0$. On each execution Step (5a1) uses the *detect* operation to test if tube T_1 contains any DNA strand. If a “yes” is returned, then Step (5b) will be run. Otherwise, that step will not be executed. Next, each execution of Step (5b) uses the *append-head* operations to append $n_{o+1,q}^1$ and $b_{o,q}^0$ onto the head of every strand in T_1 . Each execution of Step (5b1) applies the *detect* operation to examine whether tube T_2 contains any DNA strand. If a “yes” is returned, then Step (5c) will be run. Otherwise, that step will not be

executed. On each execution of Step (5c), it applies the *append-head* operations to append n_{o+1,q^0} and b_{o,q^0} onto the head of every strand in T_2 . Then, each execution of Step (5d) applies the *merge* operation to pour tubes T_1 and T_2 into $T_0^<$. Tube $T_0^<$ contains the strands finishing compare, shift, and subtract operations of a bit. Repeat execution of Steps (5a–5d) until the $(2 * k)$ bits are processed. Tube $T_0^<$ contains the strands finishing compare, shift, and subtract operations of the $(2 * k)$ bits for the o^{th} compare, shift, and subtract operations to the comparative result of less than (“<”).

Next, each execution of Step (6) applies the *merge* operation to pour tubes $T_0^{>=}$ and $T_0^<$ into T_0 . Tube T_0 contains the strands finishing the o^{th} compare, shift, and subtract operations of $(2 * k)$ bits for the comparative results of greater than or equal to or less than. Repeat execution of the steps above until successive compare, shift, and subtract operations of at most $(2 * k)$ times are processed. Tube T_0 contains the strands finishing a division for a dividend of $(2 * k)$ bits and a divisor of d bits for $1 \leq d \leq k$.

13.3.8 Finding Two Large Prime Numbers

The following DNA algorithm is applied to find two large prime numbers of k bits.

Algorithm 1: Finding two large prime numbers.

```

(1) InitialSolution( $T_0$ ).
(2) InitialProduct( $T_0$ ).
(3) For  $d = 1$  to  $k$ 
    (3a)  $T_0 = +(T_0, m_k - d + 1^1)$  and  $T_{off} = -(T_0, m_k - d + 1^1)$ .
    (3b) BinaryParallelDivider( $T_0, d$ ).
    (3c) For  $q = 1$  to  $k - d + 1$ 
        (3d)  $T_0 = +(T_0, n_k + d + 1, q^0)$  and  $T_{bad} = -(T_0, n_k + d + 1, q^0)$ .
        (3e) Discard( $T_{bad}$ ).
        (3f) If (Detect( $T_0$ ) = “no”) then
            (3g) Terminate the execution of the second (inner) loop.
        EndIf
    EndFor
    (3h) If (Detect( $T_0$ ) = “yes”) then
        (3i) Read( $T_0$ ) and then terminate the algorithm.
    EndIf
    (3j)  $T_0 = \cup(T_0, T_{off})$ .
EndFor
EndAlgorithm

```

On the execution of Step (1), it calls InitialSolution(T_0) to construct solution space of DNA strands for every unsigned integer of k bits. This means that tube T_0 includes strands encoding 2^k different integer values. Next, the execution of Step (2) calls InitialProduct(T_0) to append DNA sequences of encoding n , the product of two large prime numbers of k bits, onto the head of every strand in tube T_0 . This

implies that the front $(2 * k)$ bits and the last k bits of every strand in T_0 represent the dividend and the divisor respectively of a division instruction after Step (2) is performed.

Step (3) is two level loops and is used mainly to factor the product of two large prime numbers of k bits. On each execution of Step (3a), it uses the *extract* operation to form two tubes: T_0 and T_{off} . The first tube T_0 includes all of the strands that have $m_{k-d+1} = 1$. This is to say that the $(k-d+1)^{th}$ bit of every divisor in T_0 is equal to one. The second tube T_{off} consists of all of the strands that have $m_{k-d+1} = 0$. This indicates that the $(k-d+1)^{th}$ bit of every divisor in T_{off} is equal to zero. Because the front d bits of every divisor in T_{off} are all zeros, therefore, the d^{th} division instruction is not applied to compute the remainder of every strand in T_{off} . Next, each execution of Step (3b) calls BinaryParallelDivider(T_0, d). The procedure is used to finish a division instruction. After Step (3b) is performed, the remainder of every strand in T_0 is computed. Step (3c) is the inner loop and is mainly employed to judge whether the remainder of a division operation is equal to zero. On each execution of Step (3d), it uses the *extract* operation to form two tubes: T_0 and T_{bad} . The first tube T_0 includes all of the strands that have $n_k + d + 1, q = 0$. This means that the q^{th} bit of every remainder in T_0 is equal to zero. The second tube T_{bad} consists of all of the strands that have $n_k + d + 1, q = 1$. This implies that the q^{th} bit of every remainder in T_{bad} is equal to one. Since the strands in T_{bad} encode every remainder that is not equal to zero, Step (3e) is used to discard T_{bad} . Then, each execution of Step (3f) applies the *detect* operation to check whether tube T_0 contains any DNA strand. If a “no” is returned, then this indicates that all of the remainders in T_0 for the d^{th} division operation are not equal to zero. Therefore, Step (3g) is employed to terminate the execution of the inner loop. If a “yes” is returned, then repeat the steps until the number of the execution of the inner loop is performed.

After the inner loop is performed, Step (3h) is applied to detect whether T_0 contains any DNA strands. If it returns a “yes,” then DNA sequences in T_0 represent the remainders that are equal to zero. Hence, Step (3i) is used to find the answer (one of two large prime numbers) from T_0 . Simultaneously, the algorithm is terminated. If it returns a “no,” then Step (3j) is employed to pour tube T_{off} into tube T_0 . That is, T_0 reserves the strands that have $m_{k-d+1} = 0$. Repeat the steps until the number of the execution of the outer loop is performed. Finally, the strands in T_0 encode every strand that is zero. This indicates that the only two large prime numbers of k bits are in T_0 . Therefore, it is inferred that the difficulty of factoring the product of two large prime numbers of k bits is solved from those steps in Algorithm 1.

13.3.9 Breaking the RSA Public-Key Cryptosystem

The RSA public-key cryptosystem can be used to encrypt messages sent between two communicating parties so that an eavesdropper who overhears the encrypted message will not be able to decode them. Assume that the encrypted message overheard is represented as C (the corresponding cipher text). An eavesdropper only needs to use the following algorithm to decode them.

Algorithm 2: Breaking the RSA Public-key Cryptosystem.

(1) Call Algorithm 1.

- (2) Compute the secret key d , from the multiplicative inverse of e , module $(p - 1) * (q - 1)$ on a classical computer.
- (3) Decode the messages overheard through the decryption function, $C^d \text{ (module } n)$, on a classical computer.

EndAlgorithm

On the execution of Step (1), it calls Algorithm 1 to factor the product of two large prime numbers through three DNA-based algorithms: parallel comparator, parallel subtractor, and parallel divider. After the product is factored, computing the secret key and decoding an encrypted message are performed on a classical computer. From the steps in Algorithm 2, an eavesdropper can decode the encrypted message overheard.

13.3.10 The Complexity of Algorithm 1

Suppose that the length of n , the product of two large prime numbers of k bits is $(2 * k)$ bits. In Algorithm 1, we have found that:

1. Based upon the number of biological operations, the difficulty of factoring n can be solved with $O(k^3)$ biological operations solution space of DNA strands.
2. Based upon the number of DNA strands, the difficulty of factoring n can be solved with $O(2^k)$ library strands from solution space of DNA strands.
3. Based upon the usage of the number of tubes, the difficulty of factoring n can be solved with $O(1)$ tubes from solution space of DNA strands.
4. Based upon the longest length of DNA strands, the difficulty of factoring n can be solved with the longest library strand, $O(k^2)$, from solution space of DNA strands.

13.4 Conclusion

A general digital computer contains mainly the CPU and memory. The main function of the CPU is to perform mathematical computational tasks and the main function of memory is to store each data needed for mathematical computational tasks. However, on a general molecular computer, each data needed for mathematical computational tasks is encoded by means of a DNA strand, and performing mathematical computational tasks is by means of a DNA algorithm, including a series of basic biological operations, on those DNA strands.

This chapter presents a breakthrough biomolecular algorithm, Algorithm 1, for solving the problem of factoring. In the beginning, good DNA sequences, used to construct the solution space of DNA strands, were selected to decrease the error rate for hybridization. Second, the basic biological operations used in the Adleman-Lipton model have been performed in a fully automated manner in their laboratory. Full automation is essential not only for speeding up computation but also for error-free computation. Third, algorithm 1 contains three DNA-based algorithms to factor the product of two large prime numbers: parallel comparator, parallel subtractor, and

parallel divider. The complexity of Algorithm 1 (based upon the usage of the number of tubes, the longest length of DNA strands, the number of DNA strands, and the number of biological operations) are $O(1)$, $O(k^2)$, $O(2^k)$, and $O(k^3)$ respectively. It only takes polynomial time to factor the number of binary digits of the product (integer). After the product is factored, decoding an encrypted message is performed on a classical computer.

It is clear that molecular computing has the ability to perform complicated mathematical operations. Factoring and prime numbers are used in one of the most commonly used public-key cryptosystems. No method can be applied to break the RSA cryptosystem in a reasonable time. However, this seems to be incorrect on a molecular computer. This chapter proposes the first example of *molecular cryptanalysis* for cryptosystems based on public key, therefore demonstrating that biomolecular computing is a technology worth pursuing.

References

- [1] Rivest, R. L., A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, Vol. 2, No. 2, 1978, pp. 120–126.
- [2] Merkle, R. C., "Secure communications over insecure channels," *Commun. ACM*, Vol. 21, 1978, pp. 294–299.
- [3] Diffie, W., and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory (IEEE TIT)*, Vol. 22, 1976, pp. 644–654.
- [4] Feynman, R. P., *In Miniaturization*, D. H. Gilbert (ed.), New York: Reinhold Pub., 1961.
- [5] Adleman, L. M., "Molecular computation of solutions to combinatorial problems," *Science*, Vol. 266, 1994, pp. 1021–1024.
- [6] Lipton, R. J., "DNA solution of hard computational problems," *Science*, Vol. 268, 1995, pp. 542–545.
- [7] Roweis, S., et al., "A sticker-based model for DNA computation," *J. Computat. Biol.*, Vol. 5, 1998, pp. 615–629.
- [8] Sinden, R. R., *DNA Structure and Function*. San Diego: Academic Press, 1994.
- [9] Adleman, L. M., "On constructing a molecular computer," in *DNA Based Computers, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 27, R. J. Lipton and E. B. Baum (eds.), Amer. Math. Soc., 1996, pp. 1–21.
- [10] Paun, G., G. Rozenberg, and A. Salomaa, *DNA Computing: New Computing Paradigms*. Berlin/New York: Springer, 1998.
- [11] Watson, J. D., et al., *Recombinant DNA*, 2nd ed. New York: Scientific American Books, distributed by W.H. Freeman, 1992.
- [12] Watson, J. D., et al., *Molecular Biology of the Gene*, 4th ed. Menlo Park, CA: Benjamin-Cummings, 1987.
- [13] Blackburn, G. M., and M. J. Gait, *Nucleic Acids in Chemistry and Biology*, 2nd ed., Oxford/New York: Oxford Univ. Press, 1996.
- [14] Eckstein, F., *Oligonucleotides and Analogues: A Practical Approach*. Oxford/ New York: IRL Press, 1991.
- [15] Braich, R. S., et al., "Solution of a satisfiability problem on a gel-based DNA computer," in *Proc. 6th Intl. Conf. DNA Computation in the Springer-Verlag Lecture Notes in Computer Science Series*, Vol. 2054, 2000, pp. 27–42.
- [16] Braich, R. S., et al., "Solution of a 20-variable 3-SAT problem on a DNA computer," *Science*, Vol. 296, 2002, pp. 499–502.

- [17] Narayanan, A., and S. Zorbalas, "DNA algorithms for computing shortest paths," in *Genetic Programming 1998: Proc. Third Ann. Conf. (GP-98)*, Univ. Wisconsin, Madison, 1998, pp. 718–723.
- [18] Chang, W.-L., and M. Guo, "Solving the dominating-set problem in Adleman-Lipton's model," in *Proc. 3rd Intl. Conf. Parallel and Distributed Computing, Applications and Technologies (PDCAT '02)*, Kanazawa Bunka Hall, Kanazawa, Japan, 2002, pp. 167–172.
- [19] Chang, W.-L., and M. Guo, "Solving the clique problem and the vertex cover problem in Adleman-Lipton's model," Presented at *Proc. IASTED Intl. Conf. Networks, Parallel and Distributed Processing, and Applications*, Tsukuba, Japan, 2002.
- [20] Chang, W.-L., and M. Guo, "Solving NP-complete problem in the Adleman-Lipton model," in *Proc. 3rd Intl. Conf. Computer and Information Technology*, Japan, 2002, pp. 157–162.
- [21] Chang, W.-L., and M. Guo, "Resolving the 3-dimensional matching problem and the set packing problem in Adleman-Lipton's model," in *Proc. IASTED Intl. Conf. Networks, Parallel and Distributed Processing, and Applications*, Tsukuba, Japan, 2002, pp. 431–436.
- [22] Chang, W.-L., and M. Guo, "Solving the set cover problem and the problem of exact cover by 3-sets in the Adleman-Lipton model," *Biosystems*, Vol. 72, 2003, pp. 263–275.
- [23] Boneh, D., C. Dunworth, and R. J. Lipton, "Breaking DES using a molecular computer," Princeton Univ. Tech. Rep. CS-TR-489-95, 1995.
- [24] Bach, E., et al., "DNA models and algorithms for NP-complete problems," in *Proc. Eleventh Ann. IEEE Conf. Computational Complexity*, Philadelphia: IEEE Computer Society Press, 1996, pp. 290–300.
- [25] Fu, B., "Volume bounded molecular computation," in *Computer Science*, New Haven, CT: Yale Univ., 1997, p. 87 (Ph.D. thesis).
- [26] Ouyang, Q., et al., "DNA solution of the maximal clique problem," *Science*, Vol. 278, 1997, pp. 446–449.
- [27] Shin, S.-Y., B.-T. Zhang, and S.-S. Jun, "Solving traveling salesman problems using molecular programming," in *Proc. Congress on Evolutionary Computation (CEC99)*, Vol. 2, Washington, DC, 1999, pp. 994–1000.
- [28] Arita, M., A. Suyama, and M. Hagiya, "A heuristic approach for Hamiltonian path problem with molecules," in *Genetic Programming 1997: Proc. Second Ann. Conf. (GP-97)*, Stanford Univ.: Morgan Kaufmann Pub., 1997, pp. 457–462.
- [29] Morimoto, N., M. Arita, and A. Suyamayi, "Solid phase DNA solution to the Hamiltonian path problem," in *Proc. 3rd DIMACS Workshop on DNA Based Computers*, Pennsylvania, 1997, pp. 83–92.
- [30] Amos, M., "DNA computation," in *Computer Science*, Univ. Warwick, UK, 1997 (Ph.D. thesis).
- [31] Reif, J. H., T. H. LaBean, and N. C. Seeman, "Challenges and applications for self-assembled DNA nanostructures," in *Proc. 6th Intl. Workshop on DNA-Based Computers in the Springer-Verlag Lecture Notes in Computer Science Series*, Vol. 2054/2001, A. Condon and G. Rozenberg (eds.), Leiden, The Netherlands, 2000, pp. 27–42.
- [32] LaBean, T. H., E. Winfree, and J. H. Reif, "Experimental progress in computation by self-assembly of DNA tilings," in *Proc. 5th DIMACS Workshop on DNA Based Computers, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 54, Cambridge, MA: MIT, 1999, pp. 123–140.
- [33] Mao, C., et al., "Logical computation using algorithmic self-assembly of DNA triple-crossover molecules," *Nature*, Vol. 407, 2000, pp. 493–496.
- [34] Guarnieri, F., M. Fliss, and C. Bancroft, "Making DNA add," *Science*, Vol. 273, 1996, pp. 220–223.
- [35] Gupta, V., S. Parthasarathy, and M. J. Zaki, "Arithmetic and logic operations with DNA," in *Proc. 3rd DIMACS Workshop on DNA-based Computers*, Philadelphia, 1997, pp. 212–220.

- [36] Qiu, Z. F., and M. Lu, "Arithmetic and logic operations for DNA computers," in *Proc. 2nd IASTED Intl. Conf. Parallel and Distributed Computing and Networks*, Brisbane, Australia, 1998, pp. 481–486.
- [37] Ogihara, M., and A. Ray, "Simulating Boolean circuits on a DNA computer," Univ. Rochester, Tech. Rep. TR631, Aug. 1996.
- [38] Amos, M., and P. E. Dunne, "DNA simulation of Boolean circuits," Univ. Liverpool, UK, Tech. Rep. CTAG97009, Dec. 1997.
- [39] Atanasiu, A., "Arithmetic with membranes," in *Proc. Workshop on Multiset Processing*, Romania, 2000, pp. 1–17.
- [40] Frisco, P., "Parallel arithmetic with splicing," *Romanian J. Info. Sci. Technol. (ROMJIST)*, Vol. 3(2), 2000, pp. 113–128.
- [41] Hug, H., and R. Schuler, "DNA based parallel computation of simple arithmetic," in *Proc. 7th Intl. Workshop on DNA Based Computers in the Springer-Verlag Lecture Notes in Computer Science Series*, Vol. 2340/2002, Tampa, FL, 2001, pp. 159–166.
- [42] Barua, R., and J. Misra, "Binary Arithmetic for DNA Computers," in *Proc. 6th Intell. Conf. DNA Computation in the Springer-Verlag Lecture Notes in Computer Science Series*, Vol. 2568/2003, Sapporo, Japan, 2002, pp. 124–132.
- [43] Adleman, L. M., et al., "On applying molecular computation to the data encryption standard," *J. Computat. Biol.*, Vol. 6, 1999, pp. 53–63.
- [44] Pérez-Jiménez, M. J., and F. Sancho-Caparrini, "Solving knapsack problems in a sticker based model," in *Proc. 7th DIMACS Workshop on DNA Based Computers, DIMACS Series in Discrete Mathematics and Theoretical Computer Science; also see in Lecture Notes in Computer Science (LNCS) 2340 Springer 2002*, Tampa, FL, 2001, pp. 161–171.
- [45] Guo, M., et al., "Is optimal solution of every NP-complete or NP-hard problem determined from its characteristic for DNA-based computing," *Biosystems*, Vol. 80, 2005, pp. 71–82.
- [46] Chang, W.-L., M. Guo, and M. Ho, "Towards solution of the set-splitting problem on gel-based DNA computing," *Future Generation Computer Systems*, Vol. 20, 2004, pp. 875–885.
- [47] Chang, W.-L., M. S.-H. Ho, and M. Guo, "Molecular solutions for the subset-sum problem on DNA-based supercomputing," *Biosystems*, Vol. 73, 2004, pp. 117–130.
- [48] Guo, M., M. S.-H. Ho, and W.-L. Chang, "Fast parallel molecular solution to the dominating-set problem on massively parallel bio-computing," *Parallel Computing*, Vol. 73, 2004, pp. 1109–1125.
- [49] Cormen, T. H., C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, Cambridge, MA/New York: MIT Press and McGraw-Hill, 1990.
- [50] Garey, M. R., and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: W. H. Freeman, 1979.

Chemotaxis: Learning Navigation and Source Localization Strategies from Biology's Engineered Designs

Gail Rosen and Paul Hasler

14.1 Introduction

Everything has an odor signature. Humans can smell the chemical presence of volatile compounds in the air, but animals, with their more sensitive noses, can detect the presence of substances that are odorless to humans. For example, APOPO [1], a landmine removal organization, traces explosive vapor emanating from landmines by using the extreme sensitivity of the rat's nose. This acute sense of smell can be attributed to the fact that rats and dogs have more chemoreceptors and more developed olfactory bulbs than humans; dogs have 20 times more odor receptors than humans. Biology provides efficient mechanisms for completing difficult tasks, such as olfaction.

This leads us to the pivotal questions: how is biology efficient at chemical tracking and what principles can we learn from biology to help us with engineering design? Currently, mammalian olfaction is extremely complex and, while many components are known, our understanding of the mechanism only scratches the surface. On the other hand, there are many studies of single-celled organisms using chemotaxis, or mobilization in chemical gradients. This chapter examines chemotaxis and navigational techniques inspired by these mechanisms.

At the most fundamental level, chemical tracking is essential to primitive organisms. Humans have five senses, some highly evolved, whereas single-celled organisms essentially have two: touch and smell/taste. Without either, the organism would not be able to “hunt” its food and eat, or avoid predators. Thus, a single-celled organism must perform the computation necessary to achieve survival by integrating its senses in chemotaxis. This chapter examines chemotaxis random-walk locomotion and receptor clustering, and the associated algorithms inspired by these mechanisms:

1. A single-sensor biased random walk and a two-sensor directional sensing algorithm for gradient tracking.

2. Multiple biased random walks for tracking multiple sources.
3. A multisensor mobile array for gradient tracking using a localized chemoreceptor cooperation model.

First, we review bacterial chemotaxis principles and mathematically formulate a random walk and biased random walk so the reader understands its relation to probability theory. Then, the first two random-walk algorithms are briefly discussed, to demonstrate how this strategy can be used in single-node and multinode cases. The chapter mainly focuses on the third algorithm, which is based on chemoreceptor clustering, a relatively recently discovered chemotaxis mechanism. Finally, the various parameters and performance of each strategy are compared.

14.2 Bacterial Chemotaxis Principles

Chemotaxis is the mechanism by which an organism mobilizes in a chemical gradient. A single-celled organism is known to integrate information from its receptors, or chemical sensors, to control its movement through the flagella. The behavior of bacterial chemotaxis can be characterized in two ways: (1) a run and (2) a tumble phase. This is dictated by the effect of rotation of the flagella, the motor movement, on the organism. When the counterclockwise rotation aligns the flagella into a single rotating bundle, the bacterium swims in a straight line, known as the *run* phase. When the clockwise rotation breaks the flagella bundle apart such that each flagellum points in a different direction, it causes the bacterium to change direction, known as the *tumble* or *rotational* phase. The bacterium alternates these two phases to move, using relatively straight runs interrupted by random tumbles that reorient the bacterium (illustrated by Figure 14.1). With no gradient present, the cell randomly runs and tumbles, and exhibits a random-walk behavior (see Section 14.3 for a mathematical description of a random walk). With the introduction of a gradient, the cell will start to exhibit longer runs in the direction of the gradient before tumbling and will tumble sooner if it finds it is going in an orthogonal direction to the gradient. This is due to its temporal memory, which senses if the concentration is increasing during the past movement. This behavior exhibits a biased random walk, utilized in Kadar and Virk's [2] and Dhariwal's [3] algorithms. It is thought that this biased random walk provides directionality to the organism while keeping it flexible enough to find other sources (i.e., it prevents the organism from getting caught in a local minimum) [4].

Signaling in *E. coli* chemotaxis relies upon protein phosphorylation. The key enzyme in the pathway is a histidine kinase (CheA), whose activity is modulated by binding the chemoeffector to receptors and by the level of receptor methylation [6]. Changes in receptor methylation levels result in sensory adaptation, enabling the cell to detect further changes in concentration as it swims in chemical gradients. This is similar to our visual system adjusting to low-light levels so that we can detect subtle differences. Receptor methylation also acts as a short-term memory of the chemical environment, utilized in Dhariwal et al.'s algorithm [3] and a byproduct of Rosen/Hasler's algorithm [19].



Figure 14.1 Example of a chemotaxis run and tumble trajectory, or random-walk behavior shown by 30 seconds in the life of one *Escherichia coli* K-12 bacterium swimming in an isotropic homogenous medium. The track spans about 0.1 mm, left to right. The plot shows 26 runs and tumbles, the longest run (nearly vertical) lasting 3.6 s. The mean speed is about 21 mm/s. A stereoscopic view can be seen in Berg's paper [5].

In addition, it was observed a little over ten years ago that chemotaxis receptors form clusters at cell poles in *E. coli*, a prokaryote. Since then, chemoreceptor clustering has been demonstrated in all bacteria and archaea that have been examined to date [7]. Moreover, it has recently been shown that all other chemotaxis proteins in *E. coli* localize to the cluster of receptors [8–10], thereby forming a large sensory complex. Receptor clustering plays an even greater role in eukaryotic chemotaxis. Since a eukaryotic cell is larger than a prokaryotic cell, the cell membrane receptors are used to detect a gradient across the cell, adding spatial sensing to the temporal sensing exhibited in prokaryotes. Eukaryotic chemotaxis exhibits a receptor clustering and polarization (change in cell morphology), and its response can be formulated as four major steps [17], seen in Figure 14.7. The receptors dynamically aggregate to increase sensitivity, specificity, and convergence time in localizing the direction of the chemical gradient [21]. This type of mechanism is used in Rosen and Hasler's work [19].

Besides temporal and spatial sensing being major differences in the way prokaryotic and eukaryotic cells detect gradients, there is also a difference in the number of receptors. Because eukaryotic cells are larger, there are usually more chemoreceptors in eukaryotes on average. For example, in *E. coli*, major receptors, such as those for aspartate (Tar) and serine (Tsr), are highly abundant and number several thousand molecules per cell. Minor receptors, such as those that are specific for dipeptides (Tap), ribose, and galactose (Trg), and redox potential (Aer), are much less abundant, with only a few hundred copies per cell [6]. Up to 7500 cheA dimers can be tightly packed into a two-dimensional lattice spanning around 200 nm, the observed size of polar receptor clusters [14]. A eukaryotic cell is usually 50 times the size of a prokaryotic cell; therefore, a eukaryotic cell has more receptors on average, such as *Dictyostelium discoideum*, which has 80,000 cAMP receptors

for chemotaxis [15]. In a mammalian cell, the EGF receptor generates 20,000 and 200,000 copies per cell depending on the cell type and state [16]. Biological systems use many receptors to accomplish chemical localization, and these receptors dynamically cooperate.

14.3 Mathematical Description of a Random Walk

A random walk [11] is the sum of a Bernoulli process, I_n , where I_n is an identical and independently distributed (i.i.d.) random process taking on values from set $\{0,1\}$ with probability of p for 0 and $1-p$ for 1. A Bernoulli process has the mean, $E[I_n] = p$, and the variance is $VAR[I_n] = p(1-p)$.

For a one-dimensional (1-D) random walk, a particle changes position by $+i$ or $-i$ unit every time step. A Bernoulli random process can be defined as

$$D_n = \begin{cases} i, & I_n = 1 \\ -i, & I_n = 0 \end{cases}$$

this Bernoulli random process (or outcomes of a sequence of Bernoulli random variables) is illustrated in Figure 14.2(a). D_n can be written in terms of I_n as $D_n = i(2 \cdot I_n - 1)$, thus $E[D_n] = 2iE[I_n] - i = 2ip - i$ and $VAR[D_n] = VAR[2iI_n - i] = 4i^2 VAR[I_n] = 4i^2 p(1-p)$. Let S_n be the corresponding sum process (or random walk) of D_n . The mean and variance of S_n are respectively $nE[D_n]$ and $nVAR[D_n]$. The corresponding 1-D random walk to S_n , for $i=1$, is illustrated in Figure 14.2(b).

Since these variables are independent, one can easily extend the random-walk process to two dimensions with the x and y component having the 1-D random walk. A random walk can also be generated from uniformly distributed random integers, not just a Bernoulli random variable (RV). A 2-D random walk in Figure 14.3 was simulated with equi-probable integer step sizes, i , from -10 to 10 .

Also, if on each step, the *organism* has an affinity towards the 45° angle due to higher concentration levels in that direction, and moves in this direction by $+1, +1$ (x, y) each step in addition to the random $i = \pm 10$, this 2-D random walk has a 10% bias, as shown in Figure 14.4, compared to the 0% bias in Figure 14.3.

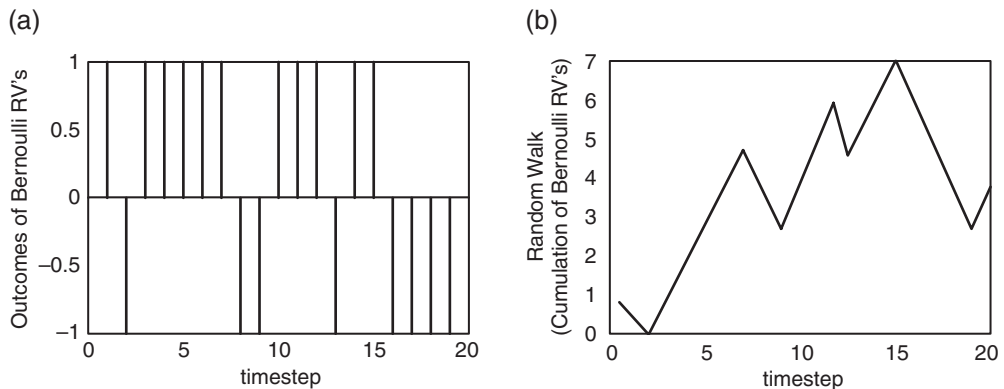


Figure 14.2 (a) Twenty outcomes of Bernoulli trials; (b) the corresponding 1-D random walk from the Bernoulli trials.

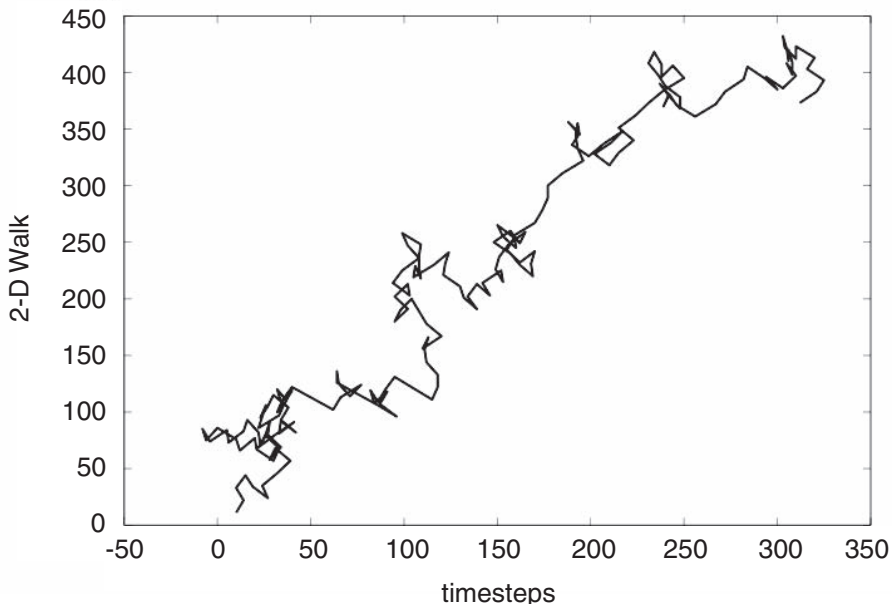


Figure 14.3 A 2-D random walk from 200 length-10 steps.

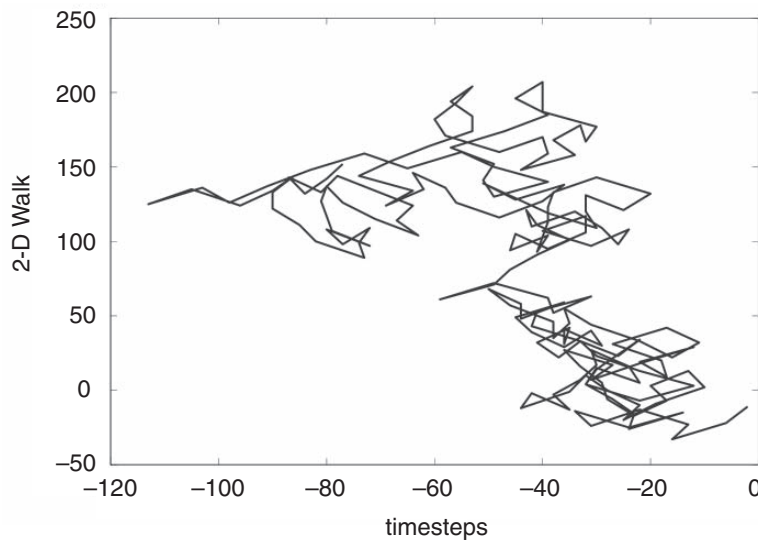


Figure 14.4 A 2-D random walk with 10% bias and 200 steps.

14.4 Chemotaxis-Based Algorithms for Diffusive Environments

The first two random-walk algorithms are briefly discussed and show how this strategy can be used in single-node and multinode cases. The chapter focuses mainly on the third algorithm, which is based on chemoreceptor clustering, a recently discovered chemotaxis mechanism.

14.4.1 Single-Node Biased Random Walk and Receptor Cooperation

Since chemotaxis is an efficient navigation strategy in gradients, engineers have designed algorithms based on this mechanism to localize diffusive sources. An initial approach was made by Kadar and Virk [2]. They compare a directional sensing algorithm they call chemotaxis to a biased random-walk model. In the terminology presented in the previous section, the biased random walk is chemotaxis movement, while their chemotaxis algorithm is a type of receptor cooperation. To keep terminology consistent, these algorithms are notated as *biased random walk* and *receptor cooperation* respectively.

The authors use a $f=1/r^2$ decay for the gradient field for the region $0 < r < 5$. The additive noise is a uniform random variable from $[-0.5, 0.5]$. All of the examples are conducted on a fixed grid composed of *units*. The *organism* is placed (4,3) units away from the source.

In the noise regimes, the initial signal-to-noise ratio (SNR) can be computed from

$$SNR = 20 \log_{10} \left| \frac{A_{signal}}{A_{noise}} \right| \quad 14.1$$

where the organism is 5 units from the source so the $A_{signal} = f = (1/5)^2$ and A_{noise} (*noise standard deviation*) = $1/\sqrt{12}$ (standard deviation for a uniform random variable). Plugging this into (14.1) produces a starting SNR of -17.17 dB.

The *biased random-walk algorithm* makes all decisions from current time measurements using a *single* sensor, and no short-term memory is assumed. For each step,

1. A run phase is executed. The run speed increases as it becomes closer to the source but slows as it homes in on the source:
 - a. More than 10 units from source, the step size is 0.5 units; thus the optimum steps to the source are 8.
 - b. Between 1 and 10 units, the step size is $0.5 + f/2$.
 - c. Less than 1 unit, the step size is $1/f$.
2. The tumble phase rotates the organism. The angle direction is the previous angle plus a uniformly chosen random variable from -28 to 28 degrees.

The *receptor cooperation algorithm* uses a fixed step size but uses the gradient between *two* sensors to gain information about the direction of the source:

1. The step size is a fixed 0.5 units, thus the optimum steps to the source are 8.
2. The positive direction of the source is computed from the two receptors on either ends of the cell (0.4 units).
3. The angle direction to progress in is chosen to be the closer of three choices, 0 or ± 14 degrees, towards the direction of the source from the line connecting the two sensors.

The results of the algorithm are summarized in Table 14.1. In a stable gradient field, (1) the receptor cooperation algorithm localizes the source directly and quickly and (2) the random-walk algorithm is indirect and slow. In the noisy field,

Table 14.1 Comparison of number of steps to source of Kadar and Virk's algorithms, averaged over five Monte Carlo runs.

	<i>Receptor Cooperation</i>	<i>Biased Random Walk</i>
Stable Field	13.4	130
Noisy Field	>1000	129.2

(1) the receptor cooperation algorithm diverges and is not likely to reach the source and (2) the random-walk algorithm performs similarly to the stable case. So while the receptor cooperation algorithm breaks down quickly in the presence of noise, the performance of the biased random-walk algorithm is the same despite the noise level.

14.4.2 Multinode Biased Random Walks for Source Tracking

Dhariwal et al. further investigates the biased random-walk aspect of chemotaxis but for the specific application of using multiple nodes to track multiple sources [3]. Rather than assuming that an organism varies its run and tumble ratio depending on the concentration level as in Kadar and Virk, Dhariwal et al. assume that the chemotaxis mechanism is based on short-term memory that is able to detect a positive or negative gradient by comparing the current concentration to the last location's concentration. This short-term memory has been verified in the biology literature [12].

On a 2000×2000 unit grid, 100 robots are placed randomly using a uniform random distribution. In biology, this can be paralleled to a colony of bacteria. The speed of each robot is assumed to progress at a unit/second and each "time step" is a second. The robot mean free path (MFP), or run length without bias, is 10 units.

The source(s) are always assumed to be a circular disc with a radius of 5 units, but two types of gradient source models are used. The first model uses m sources placed randomly on the grid and modeled by an inverse square law:

$$Intensity(x, y) = \frac{1}{K} \sum_{i=0}^m \frac{q_i}{r_i^2} \quad 14.2$$

The intensity can be sensed at a point (x, y) on the grid in the presence of m gradient sources, q_i is the intensity of the source S_i , K is a constant of proportionality, and r_i is the distance between the grid point (x, y) and the center of source S_i .

The second source model assumes that the source decays over time, such as an impulse source with infinite boundaries or actual consumption of the source where the chemical is a nutrient that can be eaten by the "bacterium" nodes. The intensity of the source, S_i , at any time t is given by

$$q_i(t) = [q_i(0)e^{-k_1 t}] - k_2 \sum_{j=0}^t N_{ij} \quad 14.3$$

where $q_i(0)$ is the initial intensity of S_i , k_1 and k_2 are constants that depend on the type of source, N_{ij} is the number of robots at source, S_i , depleting its energy at time

j. This is used in conjunction with (14.2) to create an intensity map based on decaying.

The run-and-tumble strategy used by each robot has three phases: move bias length in previous direction, tumble, run, and repeat. It can be described with the following pseudocode (each run time limit is 5×10^4 seconds and 10^4 Monte Carlo runs were averaged to get the final convergence results):

```

WHILE NOT at Gradient Source OR Time limit
  IF [(Concentration_new AND Concentration_old exist) AND
    Concentration_new > Concentration_old]
    bias length = bias * MFP;
    MOVE bias length in previous direction;
  END
  Concentration_old =  $I(x, y)$ ;
  tumble = random direction from choice of eight neighboring grid
    points, {angles: 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°};
  FOR 1 to runlength
    MOVE to next point on grid in the tumble direction;
    time step = time step + 1;
  END
  Concentration_new =  $I(x, y)$ ;
END

```

In Kadar and Virk, the bias is based on the concentration level at the current time. In Dhariwal, there is a bias if the concentration is positive (determined from a short-term memory), and the actual concentration intensity does not affect the bias.

In Figure 14.5, a scenario is run for the 100 nodes placed 900 units away from a single source. With no bias, there is little progress after 50,000 sec, but with just 10% bias, every node is able to localize the source within 40,000 sec, and 80% of the nodes reach the source within 25,000 sec. With a 40% bias, 80% are able to reach the node in 5,000 sec.

The 100 nodes are also tested for finding multiple sources. It is unknown how distant these sources are from each other, but they are introduced at different times with the same amplitude, and it takes about 5,000 sec for 10% of the nodes to reach each one after it activates and quickly decays.

Also, an error is placed on the gradient decision function to see how performance would degrade. In all biases, the gradient measurement is subject to a percentage of error (e.g., for the 6% error case, if the gradient is positive, there is a 6% chance it will be measured as negative). In this scenario, the nodes still converge to the source but at a slower rate. The 20% error case takes about 50,000 sec for all nodes to localize the source as opposed to the no-error case of 40,000 sec. So, for full convergence, it takes about 20% more time to converge. A similar trend is seen with the 40% error case, and it is expected to take around 40% longer to converge fully.

The single-source case is also expanded to a disc of 45 units, and the algorithm is shown to perform well for boundary detection. These multinode algorithms are useful for mapping chemical plumes, spills, and multiple sources.

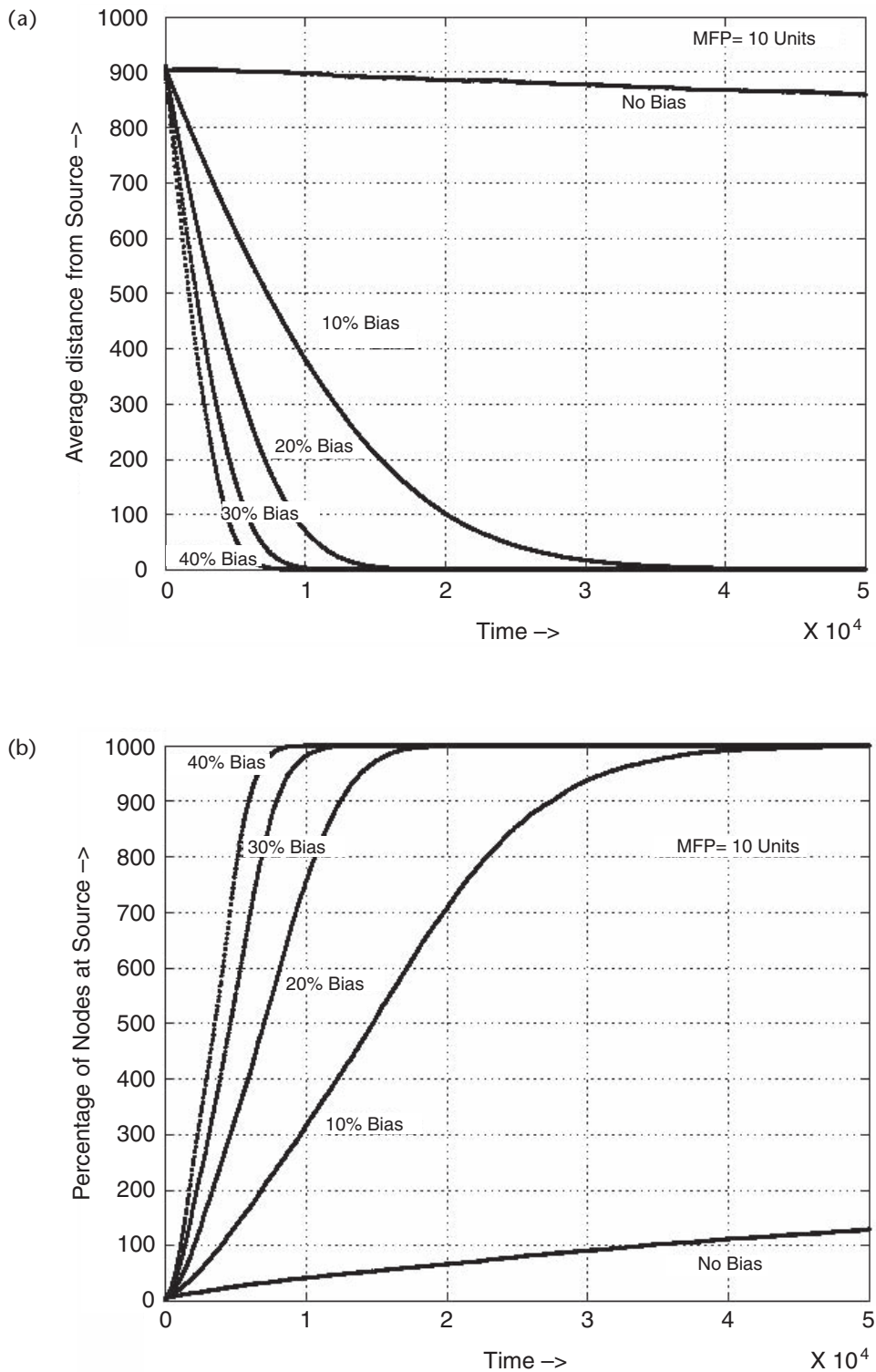


Figure 14.5 Increasing the bias decreases the time to convergence for this algorithm shown in (a) the average distance between the robots and the source vs time, and (b) the percentage of robots at the source vs time. Note there is just an inverse relationship between the two [3].

14.4.3 Multichemoreceptor Cooperation for Gradient Tracking

In Kadar and Virk's work, two algorithms based on chemotaxis are examined: the receptor cooperation and the random walk behavior. In heavy noise, the receptor cooperation algorithm quickly breaks down, while the random walk performance stays the same. But the previous receptor cooperation algorithm only utilizes two sensors spaced on opposite sides of the cell. A natural extension is to examine many receptors and their spatial sensing in eukaryotic chemotaxis, since receptor cooperation plays an important role [6]. Rosen and Hasler developed a technique based on chemoreceptor clustering exhibited in eukaryotic cells [19].

For practical reasons, it is unfeasible to develop a multireceptor system with as many receptors as in biology (e.g., the 20,000 and 200,000 copies per mammalian cell). As shown in Section 14.2, several receptors can enhance the localization with a clustering behavior. In this section, an algorithm based on the receptor clustering behavior [19] is shown to improve fixed sensor array performance in navigating chemical gradients.

14.4.3.1 Hebbian Learning for Adaptation

To simulate a localized receptor clustering algorithm, a form of neural learning is used to adapt the neural weights behind the receptors. Hebb's rule, a classical learning technique that adapts a set of weights to the input signal, is used. For the goal at hand, the algorithm learns the connection strengths (correlations) between sensors to determine what direction a source is coming from. A sensor that has higher correlation than other sensors is one that gets a higher amplitude on input and is therefore closer to the source.

A discrete-time matrix form of the Hebbian learning rule can be expressed as

$$\begin{aligned}\mathbf{W}[n+1] &= \mathbf{W}[n] + \eta \mathbf{R}_{xx}[n] \mathbf{W}[n] \\ &= (\mathbf{I} + \eta \mathbf{R}_{xx}[n]) \mathbf{W}[n]\end{aligned}$$

where $\mathbf{x}[n]$ is a vector of N inputs at time n , $\mathbf{W}$ is a $N \times N$ matrix of adaptive weights, $\mathbf{R}_{xx}[n] = \mathbf{x}[n] \mathbf{x}^T[n]$ is the correlation of the inputs, $\mathbf{x}[n]$, and η is a constant [13].

The change in $\mathbf{W}$ over a time period is proportional to the average of the input correlation matrix

$$\Delta \mathbf{W} \sim \frac{1}{N} \sum_{n=0}^N \mathbf{R}_{xx}[n]$$

Therefore, each element, w_{ij} , can be viewed as how well the i^{th} sensor input correlates with the j^{th} sensor input. The η introduces a learning rate and short-term memory to the system. As a result, $\mathbf{W}$ can be viewed as the neural connections between each sensor and will hold memory of their connections for a short period of time. A graphical illustration of an auto-associative Hebbian network is shown in Figure 14.6. The mutual connection between sensors is analogous to sensor cooperation found in biology. The major difference is that in the Hebbian matrix adaptation, the sensors are fixed and the connections between them are adapting, while in biology, the receptor locations adapt (see Figure 14.7).

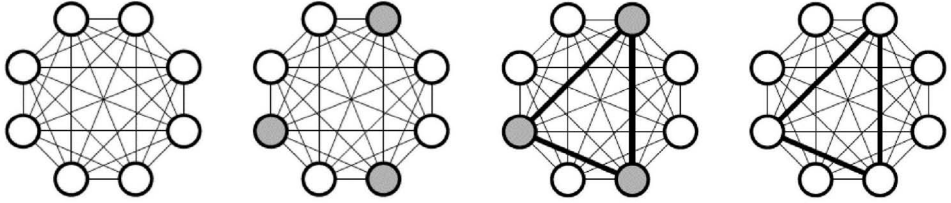


Figure 14.6 A simple Hebbian fully connected auto-associative network. When three of the units are activated by an outside stimulus, their mutual connections are strengthened. The next time the nodes are activated, they will activate each other.

14.4.3.2 Modeling a Diffusive Field

A chemical field is dynamic in nature if turbulence (due to wind, etc.) and noise (such as molecular Brownian motion) are taken into consideration. Excluding all these factors, only molecular diffusion is modeled; the concentration, C (moles/cm³), from a continuous release point source in three dimensions is the solution to Fick's Second Law [20]:

$$C(r, t) = \frac{\mu}{4\pi Dr} \operatorname{erfc} \frac{r}{2\sqrt{Dt}} \quad 14.4$$

where μ is the chemical release rate (moles/s), and D is the diffusion constant (cm²/s). The r is the radius from the point source, and t is the time from its release. In Rosen and Hasler's model, the diffusion field of interest is at long diffusion times ($t \rightarrow \infty$); therefore, the dimensionality of (14.4) is reduced to

$$C(r, t) = \frac{\mu}{4\pi Dr} \quad 14.5$$

Because the release rate varies greatly in nature and for ease of use, the expression $\mu/4\pi D$ is set to 1. Therefore, the ideal source is modeled as $C(r) = 1/r$ diffusion field. Although this is a 3-D diffusion field, the sensor array traverses a planar slice of this field.

Our sensor array is modeled as follows. Each sensor, $v_k[n]$, is the k^{th} input of N inputs at time sample n , which measures the concentration signal, $C_k(r) = 1/r_k$, at r_k away from the source. The sensors take measurements, which are contaminated with independent and identically distributed (i.i.d.) noise:

$$\mathbf{v}[n] = \mathbf{c}[n] + \mathbf{n}[n] \quad 14.6$$

where $\mathbf{c} = [C_1, C_2, \dots, C_N]^T$, $\mathbf{n} = [n_1, n_2, \dots, n_N]^T$, $\mathbf{0}$ is an $N \times 1$ vector of zeros, $\mathbf{n} \sim \text{Gaussian}(\mathbf{0}, \Sigma)$, and $\Sigma = \sigma^2 \mathbf{I}$, where $\mathbf{I}$ is an $N \times N$ identity matrix.

14.4.4.3 Multireceptor Cooperation Algorithm

In this section, a Hebbian learning method with constraints for receptor cooperation is presented, and how the constraint effects the algorithm and determination of the direction-of-arrival is described. The inputs, $\mathbf{v}$, are correlated to a weighting/

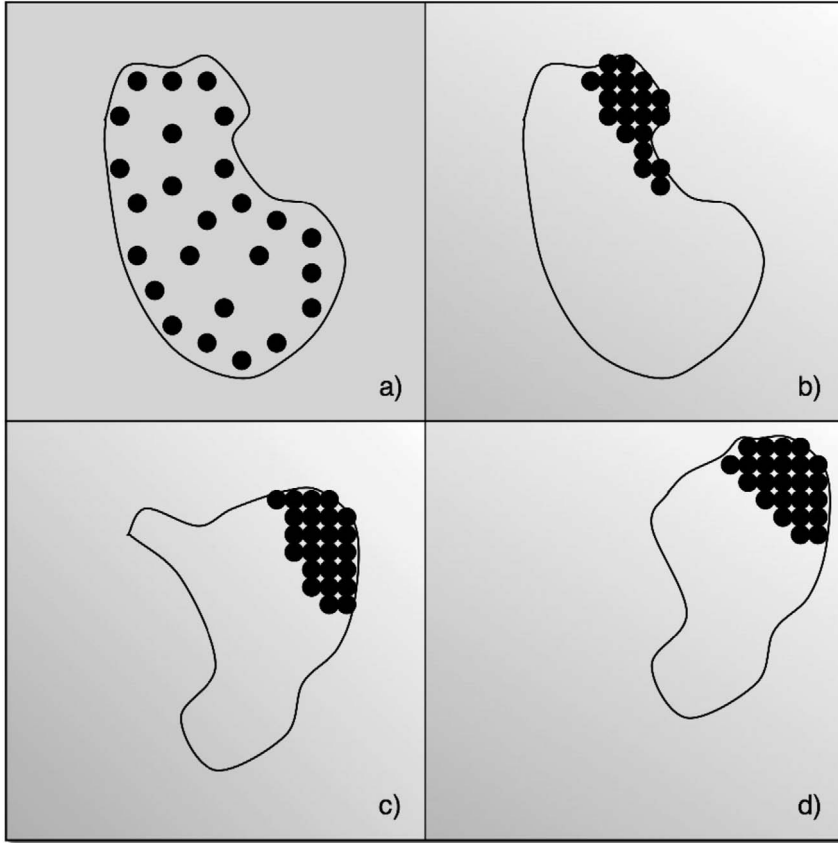


Figure 14.7 Response of a eukaryotic cell to a chemoattractant gradient. (a) With no gradient, the receptors are uniformly distributed on the cell's membrane; (b) when a gradient is introduced, the receptors cluster to the side closest to the source; (c) the cell changes its morphology to aid clustering; (d) the cell migrates towards the source.

steering matrix, $\mathbf{A}$. For each time-step iteration, n , the output of the array, $\mathbf{y}$, is computed as:

$$\mathbf{y}[n] = \mathbf{A}[n-1]\mathbf{v}[n] \quad 14.7$$

where $\mathbf{A}[0] = \mathbf{A}_{init}$ (14.10); $\mathbf{A}_{init}$ has a dual role as the initial $\mathbf{A}$ as well as constraining $\mathbf{A}$ on each iteration (14.9).

The Hebbian learning algorithm is then used to update the steering matrix:

$$\mathbf{A}[n] = \mathbf{A}[n-1] + \eta \mathbf{v}[n]\mathbf{y}^T[n] \quad 14.8$$

where $\eta = (\mathbf{v}^T[n]\mathbf{v}[n])^{-1}$. A concise view of the Hebbian algorithm with added constraint and source angle determination is discussed below and shown in Figure 14.8.

On each iteration, a constraint that controls the sensor interconnectivity is imposed on $\mathbf{A}$:

$$\mathbf{A}[n] = \mathbf{A}_{init} \circ \mathbf{A}[n] \quad 14.9$$

where $\circ$ is an element by element multiplication, and $\mathbf{A}_{init}$ is a circularly banded matrix with the band number corresponding to the sensor cooperation level, S_c :

$$\mathbf{A}_{init} = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 & \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} & 0 & 0 & \cdots & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \cdots & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ a_{N1} & 0 & \cdots & 0 & 0 & a_{NN-1} & a_{NN} \end{pmatrix}$$

$S_c = 3$ in this example means each sensor and its nearest neighbors form the output (Figure 14.7) for a direction. The connections seen in Figure 14.6 would be limited to the nearest $S_c/2 - 1$ neighbors.

This is directly related to how chemoreceptors cooperate for chemotaxis, the mechanism by which a cell senses and responds directionally to a chemical gradient. When a chemical binds to the receptors on the membrane of the cell, several receptors in a region signal a neuron. If all these receptors have chemical binds, the neuron, or weight, receives a high neural spike. Each column vector in the $\mathbf{A}$ matrix can be viewed as the neural beam pattern. It has been shown that organisms use spatial sensing mechanisms to compare receptor stimulation among different parts of the organism and then move accordingly [21]. Also, it has been observed that a cell's receptors begin to cluster towards the gradient direction when the gradient is suddenly reversed [18]. It may be due to the fact that the organism wants to increase selectivity, or its beam pattern, in that direction. We parallel this spatial clustering behavior to what is known in the array signal processing literature as beam forming [22]. So, instead of moving the sensors to increase directional selectivity, the steering matrix adapts.

The $\mathbf{A}_{init}$ is the key modification of the Hebbian learning algorithm, which limits the amount and strength of the connections, or weights in $\mathbf{A}$. Sensors closer to the source have greater impact on the computation of the source direction, while those farther away have less influence.

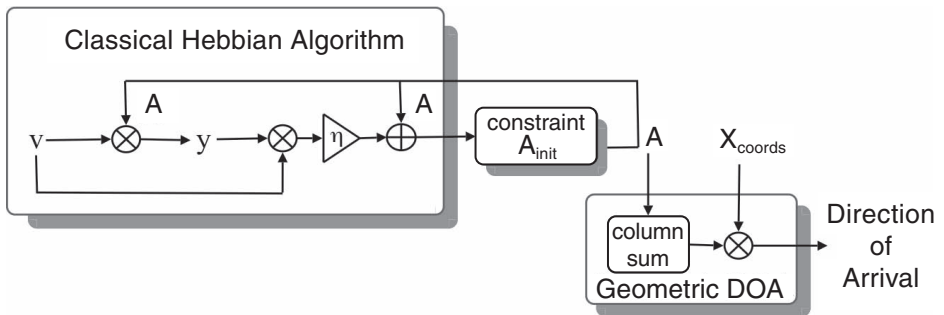


Figure 14.8 Diagram of the Hebbian learning algorithm modified for control of sensor cooperation: v are the sensor inputs, A are the adaptive weights, η is the adaptation constant, and X_{coords} are the $[x_{coords}, y_{coords}]^T$ coordinates of the sensor array. (a) Classical Hebbian learning updates the $\mathbf{A}$ matrix. (b) Each element of $\mathbf{A}$ is multiplied by each element of the constraint $\mathbf{A}_{init}$, and it restricts the amount and strengths of the sensor connectivity. (c) Each sensor's connections are summed into a total weight, which then weights the sensor coordinates to determine the direction of arrival (DOA).

Since all the connection weights to/from a sensor are summed to get the directional estimate, the constraint allows us to limit/attenuate side sensors for a particular column, which helps us control the learning algorithm's directionality and focus.

Effectively, three forms of $\mathbf{A}_{init}$ are used. *Form 1* is the case of no sensor cooperation:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

In *Form 2*, there is sensor cooperation with no side connection attenuation (example of $S_c = 3$):

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

For *Form 3*, the weights of $\mathbf{A}_{init}$ have the structure (example of $S_c = 3$):

$$\begin{pmatrix} 1 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 1 \end{pmatrix}$$

This form now places less emphasis on sensor contributions farther away from the particular focused direction (14.7). Each band S_c away from the diagonal has a $1/2^{S_c}$ weight.

Also, to keep the weights sensitive to input changes, $\mathbf{A}$ is bounded by $\eta \|\mathbf{v}\| \leq \|\mathbf{A}\| \leq \|\mathbf{v}\|$.

The direction of the sensor array movement is calculated using the steering matrix. The center of the square sensor array is (r, θ) away from the source, where

$$[r, \theta]^T = \frac{1}{N} \sum_{k=1}^N [r_k, \theta_k]^T$$

We now represent the center with Cartesian coordinates $\mathbf{x} = [x, y]^T$, and the sensor coordinates in reference to the center are

$$\mathbf{X} = \begin{bmatrix} x_1 - x & x_2 - x & \cdots & x_N - x \\ y_1 - y & y_2 - y & \cdots & y_N - y \end{bmatrix}^T = \begin{bmatrix} \mathbf{x}_{coords} \\ \mathbf{y}_{coords} \end{bmatrix}$$

Next, the direction of the source from the centroid of the array is estimated as

$$\mathbf{d}[n] = \mathbf{1}^T \mathbf{A}[n] \mathbf{X} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

where $\mathbf{1}$ is an $N \times 1$ vector of ones. The equation can be rewritten as

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} \sum_i a_{ij} \cdot \mathbf{x}_{coords} \\ \sum_i a_{ij} \cdot \mathbf{y}_{coords} \end{bmatrix}$$

where $\bullet$ is an inner product. In other words, the columns of $\mathbf{A}$ are summed, where each element in a column corresponds to a weighting of a sensor's connection to itself and other sensors. We make the assumption that the sensor with the largest summed weighting will be the closest to the source. Most likely, this is true since it receives a higher input amplitude than the other sensors. Then each summed column weights each sensor coordinate and is used to create a geometric estimate of the source direction.

The new sensor array centroid coordinate is calculated as

$$\mathbf{x}[n+1] = \mathbf{x}[n] + \delta_{fixed} \cdot \mathbf{d}[n].$$

The iteration is stopped when the center of the array is within the fixed-step threshold of the source

$$r \leq \delta_{fixed} = \frac{1}{100} r_{init}$$

A diagram of the method is seen in Figure 14.8, and an example iteration of the algorithm is shown in Figure 14.9.

When comparing this algorithm with actual chemotaxis behavior, it uses chemoreceptor clustering and not the run and tumble behavior. On the other hand, the chemoreceptor clustering relies heavily on concentration intensity, and the algorithm acts as a short-term memory (most chemotaxis methods use a short-term memory condition). Although the run and tumble algorithm is not used, the algorithm exhibits a similar trajectory to a biased random walk when in noisy conditions (Figure 14.9).

The sensor array is then simulated for a mobile scenario to evaluate the receptor cooperation results. The spatial advantage gained by a mobile, square sensor

array is assessed using three forms of steering constraint matrices (14.10): (1) an identity matrix (only one chemoreceptor for one neuron), (2) a banded matrix with unity weights (multireceptors each equally signaling a neuron), and (3) a banded matrix with $1/(2^{S_c})$ bands (multireceptors signaling a neuron, whereas receptors farther away from the neuron have less weight). Since organisms use visual cues or sensation to determine if they have reached a source, no mechanism was incorporated for the array to internally detect this. Detection is assumed when the array comes within δ_{fixed} of the source, and the steps/time for the detection to occur is the localization time. In the simulation, the center of the $3\mu\text{m} \times 3\mu\text{m}$ sensor array is placed $r_{init} = 141\mu\text{m}$ away from the source. Although the field is infinite at $(0,0)$, the source detection threshold distance is set at a fixed step size δ_{fixed} , which is $1/100$ of r_{init} . Therefore the concentration level C_{init} at the initial array placement is $1/100$ smaller than the concentration at the source threshold, δ_{fixed} .

Each sensor array is characterized by the localization time to the target vs starting SNR. Starting SNR is defined as the initial average SNR of the sensor measurements (14.6)

$$\frac{1}{N} \sum_{k=1}^N 20 \log_{10} \left| \frac{C_k[0]}{n_k[0]} \right|$$

Also, the effect of sensor cooperation on an array's localization time is assessed. The algorithm is tested over several parameters:

- N , the number of sensors, is tested over 4, 8, 16, and 32.
- S_c , the sensor cooperation level, is run for odd numbers from 1 to $N-1$.
- *starting SNR* is evaluated from approximately -8 dB to 8 dB in 2 dB steps.

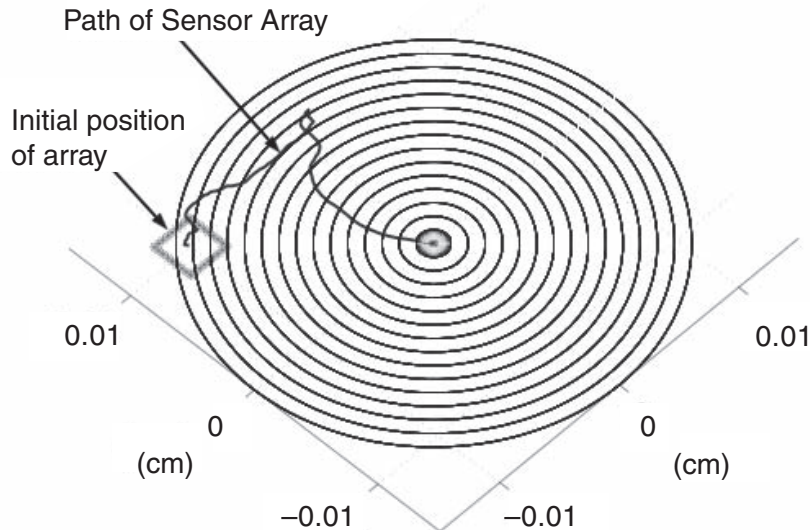


Figure 14.9 Example of navigation path of a 32-sensor array, $S_c = 5$, -1 dB starting SNR. Source location occurs in 208 steps.

The localization time is computed over all parameters; 1000 Monte Carlo iterations are computed for each combination.

In Figure 14.10, a histogram of the number of steps (or time if a velocity constant is given) for the array to reach the target is plotted. The distributions are top heavy and have very long tails. Due to limits of computation, if the number of steps exceeds 100K, the iteration is stopped and noted. In Figure 14.10, most iterations cluster around a short time value; therefore, the median is a better characterization of the heavy-tailed distributions and is the preferred statistic.

Assuming a measure-and-go strategy with δ_{fixed} as the distance the sensor array moves each step, the chemical source localizer converges to the source in 100 steps in the optimal case. As the SNR is lowered, the median number of steps to the source is used as a performance measure. For the rest of the performance measures, 10^5 Monte Carlo runs were used.

The performance gained with varying numbers of sensors and no sensor cooperation ($\mathbf{A}_{init}$ as an identity matrix) is shown in Figure 14.11. As expected, localization time increases as the SNR decreases, and it does so with a quadratic behavior. At a fixed SNR, the percentage improvement becomes less noticeable as the number of sensors increases.

To simulate an equal-weight sensor cooperation case, the $\mathbf{A}_{init}$ is set to 1 for the bands dictated by the sensor cooperation level, S_c . In Figure 14.12, the higher the sensor cooperation level for $S_c > 5$, the worse performance is. In fact, the no sensor cooperation/identity matrix case outperforms sensor cooperation for high SNR; in low SNR, $S_c = 5$ has the best overall performance. One can make sense of this by the directional pattern formed by sensor cooperation. If a sensor on the upper right-hand corner observes a high concentration coming in from the upper right, then using its nearest neighbors' (2 on each side for $S_c = 5$) measurements in addition to its own will add extra information needed to gain better resolution of the angle in that direction. On the other hand, taking all sensors around the array and weighting their information equally will cause distortion and degrade the angle resolution as opposed to using clusters in each direction.

In Figure 14.13, a comparison of the $\mathbf{A}_{init}$ forms and their influence on performance are shown for three sensor array sizes. The unequally weighted $\mathbf{A}_{init}$ (where $S_c = N/2 + 1$ for each N) performs consistently better than no sensor cooperation for all SNR. The uniform-banded $\mathbf{A}_{init}$ case (where $S_c = 3$ for $N = 8, 16$ and $S_c = 5$ for $N = 32$) is worse than no sensor cooperation for high SNR, but for low SNR this method significantly reduces localization time. A 16-sensor array using this method is comparable to a 32-sensor array with no sensor cooperation in -8dB SNR. But the two methods have trade-offs. If SNR varies, it may be more desirable to use $\mathbf{A}_{init}$ of Form 3 to consistently reduce localization time; otherwise, if the sensor array only operates in low SNR conditions, Form 2 may be more desirable.

In Table 14.2, the localization time of various sensor arrangements are numerically compared in 2 SNRs. The sensor cooperation algorithm using the third form of $\mathbf{A}_{init}$ is run for various N and S_c , and is compared to the 4-sensor, no sensor cooperation case. Increasing the number of sensors significantly improves performance, while the $\mathbf{A}_{init}$ of Form 3 helps with a localization time reduction of about 5–15%. Just by using four sensors instead of one ($N = 1$), the localization task shortens by magnitudes, as seen in Table 14.2. The single sensor case is based on a

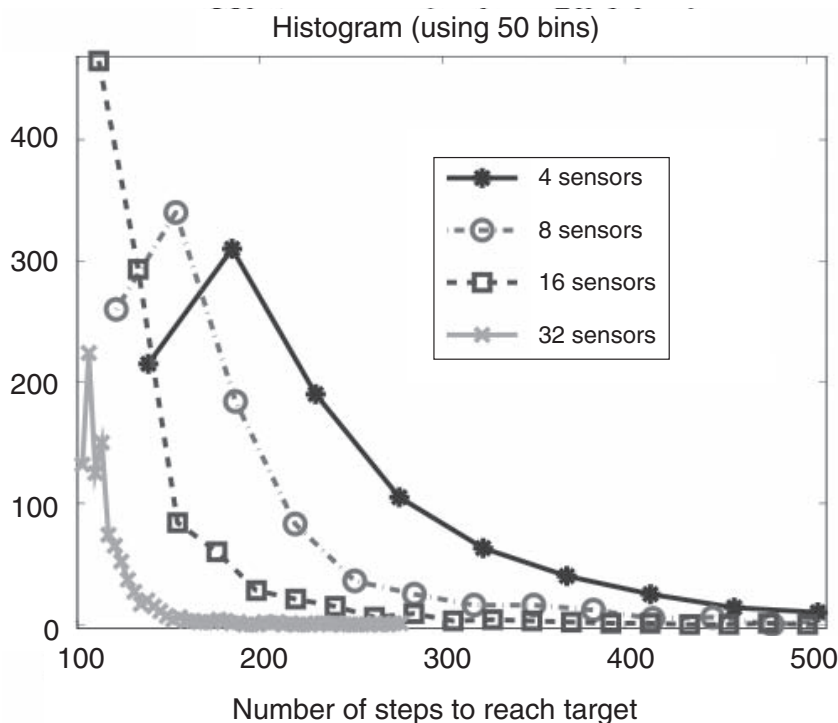


Figure 14.10 Distribution of localization time vs sensor array size for 1000 Monte Carlo runs with approximately 4 dB of sensor starting SNR and no sensor cooperation. Some tails actually extend out to around 2500 steps but are truncated for illustration.

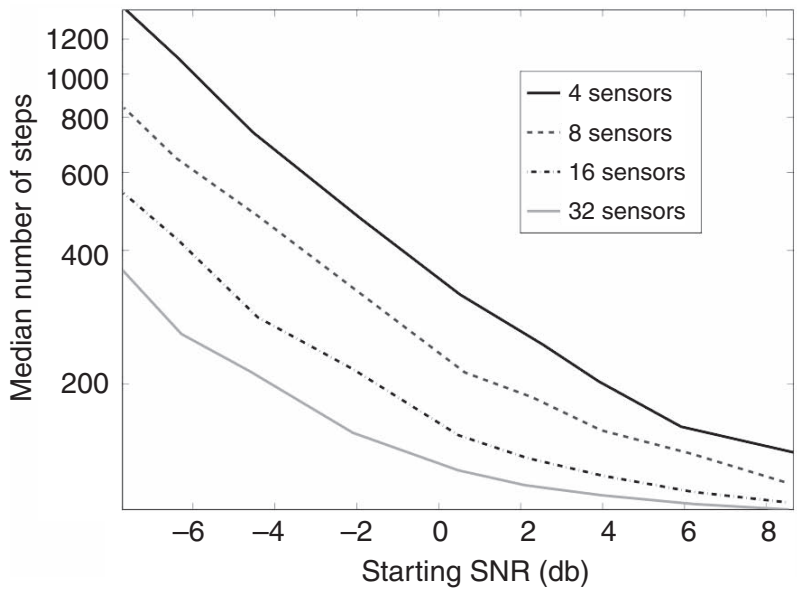


Figure 14.11 The effect of increasing the number of sensors on the localization time vs SNR; 4, 8, 16, and 32 sensors are shown, and the SNR is varied from -8 dB to 8 dB. Due to the step size, the asymptotic lower bound is 100 steps.

single sensor algorithm with memory. In the algorithm, the sensor moves randomly when the change in concentration gradient is negative; otherwise, it continues in the same direction if it is positive. The single sensor median number of steps is calculated from 50 Monte Carlo runs.

Current implementations to track heat and chemicals are underdeveloped, complicated, and/or costly. For a cost-effective solution, a small sensor array is proposed that exploits chemoreceptor cooperation to enhance performance. Eukaryotic membrane cell receptors are approximated with a square array for implementation, and the sensor array incorporates various types of sensor cooperation into the adaptive Hebbian algorithm as it tracks the source.

Noise plays a major factor in implementation due to the need to track light traces of chemicals in an environment. Simulations of a mobile array are run in various noisy conditions for three different sensor cooperation constraints: Form 1 (no sensor cooperation, classical sensor averaging), Form 2 (full-sensor cooperation), and Form 3 (a side-sensor attenuation). It is shown that sensor cooperation generally improves source localization time over the classical averaging. The Form 2 constraint sacrifices slightly poor performance at high SNR for significantly improved performance at low SNR, while the Form 3 constraint yields a consistent incremental improvement at all SNRs.

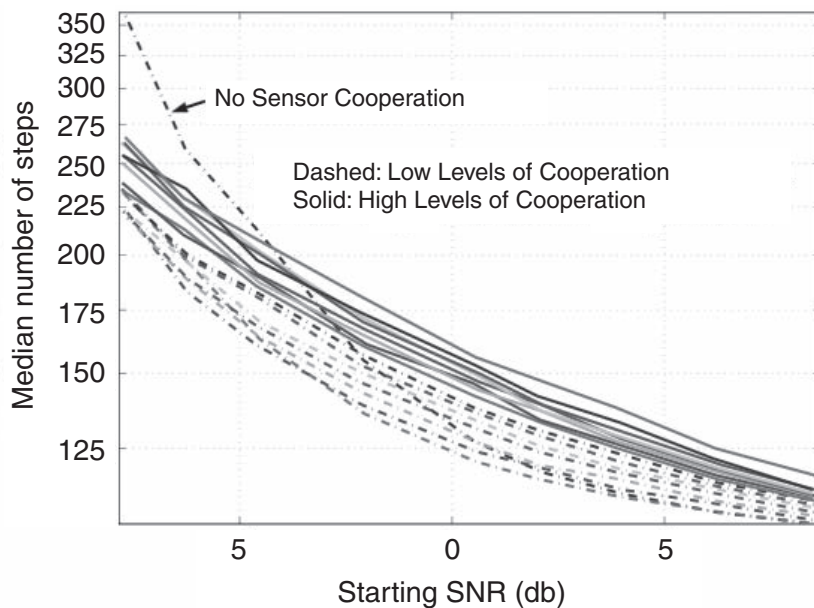


Figure 14.12 The A_{init} of Form 2 degrades performance as more sensor cooperation levels are added to a 32-sensor array. (The lower levels of sensor cooperation correspond to $\mathbf{A}$ with less than $S_c/2$ bands.) The lower levels of sensor cooperation perform better than the higher levels at all SNRs, but not as well as no sensor cooperation at high SNR. The localization time vs starting SNR is shown for the no sensor cooperation case and odd sensor cooperation levels between 3 and 31. Due to the step size, the asymptotic lower bound is 100 steps.

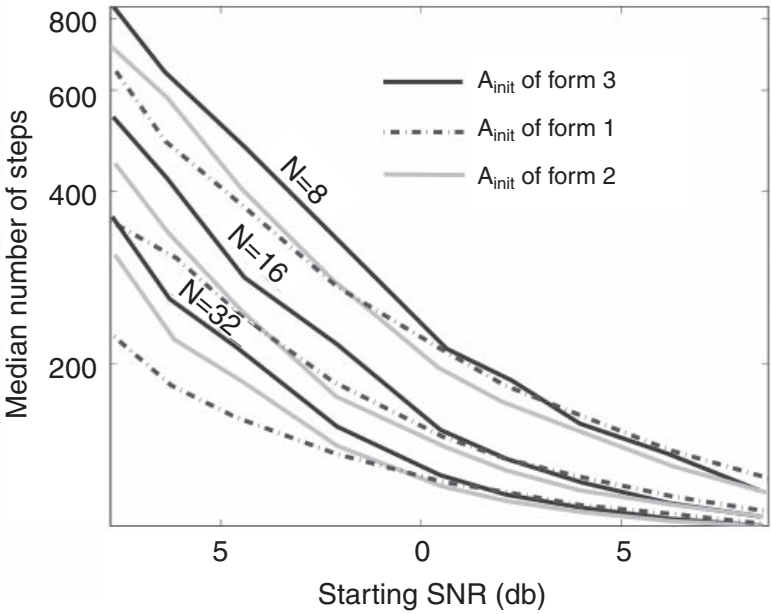


Figure 14.13 Comparison of effect localization time vs starting SNR for the three forms of A_{init} . The forms are compared for 8-, 16-, and 32-sensor arrays. Form 3 performs better than Form 1 at all SNR, while Form 2 performs much better than all algorithms at low SNR but performs slightly worse at high SNR. Due to the step size, the asymptotic lower bound is 100 steps.

14.5 Performance Comparison of the Chemotaxis Algorithms

In this chapter, three different chemotaxis-inspired algorithms are reviewed: (1) a single-node biased random-walk and receptor cooperation algorithm, (2) multinode biased random walks, and (3) a multireceptor clustering algorithm. The biased random walk is able to provide directionality while allowing enough randomness for the organism to search out a global minimum. A good example of this case is the two-source scenario (Dhariwal et al.) where various nodes are able to find multiple sources, and there is a shift of the percentage of nodes towards the larger source over time. The sensor cooperation algorithms are able to utilize the gradient infor-

Table 14.2 A comparison of the median steps (MS) for source localization at 0 dB and -7.5 dB for the single sensor mobile case, and a comparison of the $N/2+1$ banded A_{init} of Form 3 to the 4-sensor, no sensor cooperation case.

N	S_c	MS for 0 dB	MS for -7.5 dB
		(% improved)	(% improved)
1	1	11130 (-3071%)	35646 (-2511%)
4	1	351 (baseline)	1365 (baseline)
4	3	304 (13.4%)	1256 (8.0%)
8	5	210 (40.2%)	693 (49.2%)
16	9	150 (57.3%)	435 (68.1%)
32	17	126 (64.1%)	301 (77.9%)

mation directly to navigate to a source. When local groups of sensors, or receptor clusterings, are fused to spatially smooth sensor information in addition to time averaging, an array of sensors is able to perform better in a noisy environment than when each sensor adapts independently. The receptor cooperation algorithm is useful for low SNR and low gradient scenarios to exploit the directionality out of sensor inputs.

In Table 14.3, the parameters of each algorithm are categorized for comparison: the number of sensors, whether the sensors are independent or cooperative, the noise level, the optimum number of fixed steps to the source, and the number of steps to the source/localization time. With the multitudes of differences between each algorithm, it can be difficult to compare the performance between each algorithm. The step size may be variable or not exist at all if the algorithm is continuous time and not discrete time. The algorithms in this chapter use a fixed or minimum step size, so the optimum number of steps is reported, and the results are compared using a normalized time measure:

$$steps_{\text{normalized}} = \frac{steps_{\text{actual}}}{steps_{\text{optimum}}} \quad 14.14$$

This may not be the best metric of such algorithms, since the localization time is the true metric. Since these algorithms can be implemented with any node velocity, it is difficult to compare these before implementation. A standardized set of performance metrics are much needed to compare the algorithms.

The performance of these algorithms is compared in Table 14.4. The number of sensors used, the noise level, and the normalized number of steps is shown for each algorithm. An interesting note is that Kadar/Virk's algorithm does very well for a low starting SNR. It is important to note that the results were averaged only over five Monte Carlo runs, while 10^4 and 10^5 Monte Carlo runs were averaged in Dhariwal et al. and Rosen/Hasler respectively. So, the findings in Kadar/Virk may not have sufficient statistics for this result. Nonetheless, their work was one of the first navigation techniques to try both the run-and-tumble and gradient-following strategies. For the number of nodes, Dhariwal's algorithm takes much longer to converge to the source, but because of multiple nodes, the method has the advantage of finding multiple sources and even boundaries of chemicals, such as oil spills. In Rosen/Hasler, a 4-sensor array at low SNR obtained reasonable results for a single source, and this technique has the advantage of using a single node while enhancing performance via numerous sensors and algorithmic complexity. For example, in Table 14.2, by quadrupling the number of sensors with sensor cooperation in this algorithm, the localization time can decrease by three-fold.

14.6 Summary

Millions of years of evolution have resulted in lifeforms developing superior mechanisms to ensure their survival in environmental surroundings. Chemotaxis, or chemical gradient sensing, is one of the most fundamental of these mechanisms. By studying nature's designs, we can learn much about functional system design under

Table 14.3 Comparison of various parameters in each algorithm.

	<i>Number of sensors</i>	<i>Independent or cooperative nodes</i>	<i>Noise level</i>	<i>Optimum steps for fixed step size</i>
Kadar/Virk	1,2	Independent/Cooperative	−17.17 dB	8
Dhariwal et al.	100	Independent	0,6,20,40% error on binary gradient decision	90
Rosen/Hasler	4,8,16,32	Cooperative	−8 to 8 dB	100

constraints (e.g., sensing chemical gradients and locating their sources in nonideal noisy and turbulent conditions). While most prokaryotic cells accomplish this task with a temporal biased random walk strategy, eukaryotic cells gain better spatial resolution by using mobile receptors that adaptively cluster to gain better resolution of the chemical flux.

We humans have very similar chemical sensing needs, but our survival does not solely depend on our olfactory input, and therefore we do not have as refined a sense of olfaction as other mammals. Yet in today’s age, we need to locate and avoid dangerous chemicals; thus an electronic nose is needed to reduce the cost of training animals to locate explosives and other illegal substances. Currently, most electronic nose designs incorporate sensors, robotics, and signal processing but are still not usable on a wide scale. Scientists and engineers are now turning to biology to enhance these systems.

Biology’s chemotaxis designs give us a starting point to engineer systems that operate on a larger scale. The techniques presented in this chapter illustrate how chemotactic cell responses provide useful methods for chemical navigation and localization: (1) a single-node biased random walk and receptor cooperation algorithm, (2) multimode biased random walks, and (3) a multireceptor clustering algorithm. The first method is a simple one node but can diverge in noisy conditions. The second method has the advantage of multiple nodes to get a better cov-

Table 14.4 Performance comparison of the algorithms, showing the strategy, number of sensors, noise regime, and localization time normalized by the optimum step size.

	<i>Parameters</i>	<i>Normalized number of steps (14.14)</i>	
Kadar/Virk	1 sensor biased random walk, −17.17 dB starting SNR	(130/8)	16.25
Dhariwal et al.	100 sensors, 20% error (steps calculated from 50,000 seconds divided by MFP = 10)	(5,000/90)	55.56
Rosen/Hasler	4 sensors, 3 sensor cooperation, −7.5 dB starting SNR	(1,256/100)	12.56

erage area, yet may be a disadvantage for actual implementation. The third technique has the advantage of using one node with multiple sensors to get a better source localization estimate in noisy fields.

Chemotaxis is a well-studied sensing mechanism and provides a good foundation for improving our understanding of how organisms solve chemical navigation. But chemical localization approaches are not limited to just single-cell behavior. Engineers have also developed techniques inspired by moth flight strategies and cooperative flocking behavior. Developing an affordable and accurate odor localization system is a challenging problem, and engineers are learning from biology to find better solutions.

References

- [1] *Vapor Detection Technology*. Antwerp, Belgium, 2005, <http://www.apopo.org>.
- [2] Kadar, E., and G. Virk, "Field theory based navigation for autonomous mobile machines," presented at Intelligent Components for Vehicles Workshop, Portsmouth, UK, 1998.
- [3] Dhariwal, A., G. S. Sukhatme, and A. A. Requicha, "Bacterium-inspired robots for environmental monitoring," presented at IEEE International Conference on Robotics and Automation, New Orleans, LA, 2004.
- [4] Muller, S., et al., "Optimization based on bacterial chemotaxis," *IEEE Transactions on Evolutionary Computation*, Vol. 6, 2002.
- [5] Berg, H. C., and D. A. Brown, "Chemotaxis in *Escherichia coli* analysed by three-dimensional tracking," *Nature*, Vol. 239, 1972, pp. 500–504.
- [6] Sourjik, V., "Receptor clustering and signal processing in *E. coli* chemotaxis," *Trends Microbiol.*, Vol. 12, 2004, pp. 569–576.
- [7] Gestwicki, J. E., et al., "Evolutionary conservation of methyl-accepting chemotaxis protein location in bacteria and *Archaea*," *J. Bacteriol.*, Vol. 182, 2000, pp. 6499–6502.
- [8] Maddock, J. R., and L. Shapiro, "Polar location of the chemoreceptor complex in the *Escherichia coli* cell," *Science*, Vol. 259, 1993, pp. 1717–1723.
- [9] Sourjik, V., and H. C. Berg, "Localization of components of the chemotaxis machinery of *Escherichia coli* using fluorescent protein fusions," *Mol. Microbiol.*, Vol. 37, 2000, pp. 740–751.
- [10] Banno, S., et al., "Targeting of the chemotaxis methylesterase/deamidase CheB to the polar receptor-kinase cluster in an *Escherichia coli* cell," *Mol. Microbiol.*, Vol. 53, 2004, pp. 1051–1063.
- [11] Leon-Garcia, A., *Probability and Random Processes for Electrical Engineering*, 2nd ed., Menlo Park, CA: Addison-Wesley, 1994.
- [12] Foxman, E. F., E. J. Kunkel, and E. C. Butcher, "Integrating conflicting chemotactic signals: The role of memory in leukocyte navigation," *J. Cell Biol.*, Vol. 147, 1999, pp. 577–587.
- [13] Principe, J. C., N. R. Euliano, and W. C. Lefebvre, *Neural and Adaptive Systems: Fundamentals Through Simulations*, New York: John Wiley, 1999.
- [14] Li, M., and G. L. Hazelbauer, "Cellular stoichiometry of the components of the chemotaxis signaling complex," *J. Bacteriol.*, Vol. 186, 2004, pp. 3687–3694.
- [15] Ishii, D., et al., "Stochastic modelling for gradient sensing by chemotactic cells," *Sci. Tech. Adv. Materials*, Vol. 5, 2004.
- [16] Arkin, A., "Receptors across several organisms," Univ. California, Berkeley, 2006. Personal communication.

- [17] Krishnan, J., and P. A. Iglesias, "Uncovering directional sensing: where are we headed?," *IEEE Syst. Biol.*, Vol. 1, 2004, pp. 54–61.
- [18] Bray, D., M. D. Levin, and C. J. Morton-Firth, "Receptor clustering as a cellular mechanism to control sensitivity," *Nature*, Vol. 393, 1998, pp. 85–88.
- [19] Rosen, G. L., and P. E. Hasler, "Biologically-inspired odor localization using beamforming," presented at IEEE Workshop on Genomic Signal Processing and Statistics, Baltimore, MD, 2004.
- [20] Porat, B., and A. Nehorai, "Localizing vapor-emitting sources by moving sensors," *IEEE Transactions on Signal Processing*, Vol. 44, 1996.
- [21] Dusenbury, D. B., "Spatial sensing of stimulus gradients can be superior to temporal sensing for free-swimming bacteria," *Biophys. J.*, Vol. 74, 1998, pp. 2272–2277.
- [22] Johnson, D. H., and D. E. Dudgeon, *Array Signal Processing: Concepts and Techniques*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [23] Lilienthal, A., D. Reiman, and A. Zell, "Gas source tracing with a mobile robot using an adapted moth strategy," *Autonome Mobile Systeme (AMS)*, Vol. 18, 2003, pp. 150–160.
- [24] Hayes, A., "Self-organized robotic system design and autonomous odor localization," Ph.D. thesis, Pasadena, CA: California Institute of Technology, 2002.

Systems Bioinformatics: Trends and Conclusions

The discovery of DNA and its encoding provided researchers the alphabet of biological code. With the human genome project, scientists were able to obtain a manuscript of biological expressions. Current scientific work is beginning to extract the meaning behind these genes, proteins, and their interactions. The next phase will involve a transition: from scientific discovery to engineering-based design. As documented in this book, the shift is already well underway. This book illustrated how one can combine these two paradigms into an integrated framework. In Parts II–IV, analysis using engineering methods was used to learn more about biology. Part V brought design principles to biological design. The highest potential for discovery, however, lies in integration of analysis and design. One approach to integration involves using knowledge gained from biological analysis to design better engineering systems—as outlined in Part VI.

While such integrative strategies are still a nascent field, they have much potential. In the future, other strategies may be possible. For example, an iterative process can be used to simultaneously analyze, design, and refine systems in engineering using duals within biology in real-time. Such design/analysis approaches may become a reality as new automation technology makes it possible to capture system dynamics in the biological domain in real-time.

Another trend that has significant potential involves using modules to increase the abstraction level. Whereas individual gene functions are now being examined, or individual parts being designed in synthetic biology—the future may lie in analyzing/designing systems that look at combinatorial design using modules containing hundreds of genes or parts. By reducing the degrees of freedom through use of modules, this would allow for complicated systems to be feasible. Current challenges include developing standards on the definition of modules and their interconnections.

One of the key themes in this book has been the importance of integration of seemingly disparate disciplines. In order to maximize the potential of such integration, collaborations across fields are becoming imperative. For instance, this can be seen in the diverse backgrounds of the contributing authors to each chapter of this book. To encourage collaborations, correspondence information (e.g. email) is included for each chapter in the book's appendix. In addition, correspondences to the editors can be addressed to Gil Alterovitz at gil_alterovitz@hms.harvard.edu (or ga@alum.mit.edu). Lastly, readers are encouraged to visit the book's internet site at artechhouse.com for supplementary information, program code, and other resources. This is a quickly changing field—so in addition to papers, conferences are often a good place to hear the latest developments.

Best wishes in exploring the interface of engineering and biology.

Contributing Authors and Contact Information

Chapter 1: Molecular and Cellular Biology: An Engineering Perspective

Gregory Crowther

*Department of Chemical Engineering, University of Washington,
Seattle, WA, USA.*

Catherine Speake

Department of Pathobiology, University of Washington, Seattle, WA, USA.

Alicia McBride

*Department of Technical Communication, University of Washington,
Seattle, WA, USA.*

Mary Lidstrom

*Department of Chemical Engineering, University of Washington, Seattle, WA,
USA; Department of Microbiology, University of Washington, Seattle, WA, USA.*

Corresponding Author: Gregory Crowther, crowther@u.washington.edu.

Chapter 2: Proteomics: From Genome to Proteome

Stephanie Mohr, Yanhui Hu, and Joshua LaBaer

Harvard Institute of Proteomics, Harvard Medical School, Boston, MA, USA.

Corresponding Author: Joshua LaBaer, joshua_labaer@hms.harvard.edu

Chapter 3: Introduction to Biological Signal Processing at the Cell Level

Maya Said

*Department of Electrical Engineering and Computer Science, Massachusetts
Institute of Technology, Cambridge, MA, USA.*

Corresponding Author: Maya Said, mayasaid@mit.edu

Chapter 4: Signal Processing Methods for Mass Spectrometry

Peter Monchamp and Lucio Cetto

Computational Biology Development, The Math Works, Natick, MA, USA.

Jane Zhang

Biomedical Engineering Department, Boston University, Boston, MA, USA.

Rob Henson

Computational Biology Development, The Math Works, Natick, MA, USA.

Corresponding Author: Rob Henson, Rob.Henson@mathworks.com

Chapter 5: Control and Systems Fundamentals

Fulvia Ferrazzi and Riccardo Bellazzi

Department of Information and Systems Science, University of Pavia, Italy.

Corresponding Author: Fulvia Ferrazzi, fulvia@aim.unipv.it

Chapter 6: Modeling Cellular Networks

Tae Jun Lee, Chee Meng Tan, and Dennis Tu

Department of Biomedical Engineering, Duke University, Durham, NC, USA.

Lingchong You

*Department of Biomedical Engineering, Duke University, Durham, NC, USA;
Institute for Genome Sciences and Policy, Duke University, Durham, NC, USA*

Corresponding Author: Lingchong You, you@duke.edu

Chapter 7: Topological Analysis of Biomolecular Networks

Vinayak Muralidhar

*Harvard/MIT Division of Health Sciences and Technology, Massachusetts
Institute of Technology, Cambridge, MA, USA; Harvard Partners Center for
Genetics and Genomics, Harvard Medical School, Boston, MA, USA.*

Gabor Szabo

*Center for Complex Network Research, Notre Dame University, Notre Dame,
IN, USA.*

Gil Alterovitz

*Harvard/MIT Division of Health Sciences and Technology, Massachusetts
Institute of Technology, Cambridge, MA, USA; Harvard Partners Center for
Genetics and Genomics, Harvard Medical School, Boston, MA, USA; Children's*

*Hospital Informatics Program, Harvard Medical School, Boston, MA, USA;
Department of Electrical Engineering and Computer Science, Massachusetts
Institute of Technology, Cambridge, MA, USA.*

Corresponding Author: Vinayak Muralidhar, vinayakm@mit.edu

Chapter 8: Bayesian Networks for Genetic Analysis

Paola Sebastiani

Department of Biostatistics, Boston University, Boston, MA, USA.

Maria Abad-Grau

School of Computer Engineering, University of Granada, Granada, Spain.

Corresponding Author: Paola Sebastiani, sebas@bu.edu

Chapter 9: Fundamentals of Design for Synthetic Biology

Cody Wood

*Department of Biomedical Engineering, Case Western Reserve University,
Cleveland, OH, USA.*

Gil Alterovitz

*Harvard/MIT Health Science and Technology Division, Massachusetts Institute
of Technology, Cambridge, MA, USA; Children's Hospital Informatics Program,
Harvard Medical School, Boston, MA, USA; Harvard Partners Center for
Genetics and Genomics, Harvard Medical School, Boston, MA, USA;
Department of Electrical Engineering and Computer Science, Massachusetts
Institute of Technology, Cambridge, MA, USA.*

Corresponding Author: Gil Alterovitz, gil_alterovitz@hms.harvard.edu or
ga@alum.mit.edu

Chapter 10: BioJADE: Designing and Building Synthetic Biological Systems from Parts

Jonathan A. Goler

*Department of Bioengineering, University of California at Berkeley, Berkeley,
CA, USA.*

Tom Knight

*Department of Electrical Engineering and Computer Science, Massachusetts
Institute of Technology, Cambridge, MA, USA.*

Corresponding Author: Jonathan A. Goler, jagoler@mit.edu

Chapter 11: Applied Cellular Engineering

Brian M. Baynes and William J. Blake
Codon Devices, Cambridge, MA, USA.

Corresponding Author: Brian M. Baynes, bbaynes@codondevices.com

Chapter 12: The Three Faces of DNA/RNA Sequence Hybridization

Olgica Milenkovic
*Electrical and Computer Engineering, University of Colorado at Boulder,
Boulder, CO, USA.*

Corresponding Author: Olgica Milenkovic, milenkov@schof.colorado.edu

Chapter 13: Application of Biomolecular Computing to Breakthroughs in Cryptography

Michael Shan-Hui Ho
School of Information Technology, Ming Chuan University, Taiwan.

Weng-Long Chang
*Department of Computer Science and Information Engineering, National
Kaohsiung University of Applied Sciences, Taiwan.*

Minyi Guo
*School of Computer Science and Engineering, University of Aizu, Aizu
Wakamatsu City, Japan.*

Corresponding Author: Michael Shan-Hui Ho, mhoincerritos@yahoo.com

Chapter 14: Chemotaxis: Learning Navigation and Source Localization Strategies from Biology's Engineered Designs

Gail Rosen
*Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta,
GA, USA.*

Paul Hasler
*Electrical and Computer Engineering Department, Drexel University,
Philadelphia, PA, USA.*

Corresponding Author, Gail Rosen, gailr@ece.gatech.edu

About the Editors



Gil Alterovitz, Ph.D.

Gil Alterovitz, on faculty at the Harvard Medical School, is engaged in research that applies engineering systems approaches to biomedical problems. He is a part of the Children's Hospital Informatics Program at the Harvard/MIT Division of Health Sciences and Technology. He is also affiliated with the MIT Department of Electrical Engineering and Computer Science. Dr.

Alterovitz is currently heading a new class at Harvard University, "Proteomics and Cellular Network Engineering." He has served on the Harvard/MIT Division of Health Sciences and Technology M.D. Curriculum and the Harvard/MIT Division of Health Sciences and Technology Ph.D. Admission committees. He was a U.S. Fulbright to Canada (University of Toronto) in 1998–1999. He received his Ph.D. in electrical and biomedical engineering at the Massachusetts Institute of Technology through the Harvard/MIT Division of Health Sciences and Technology. Dr. Alterovitz has an S.M. degree from the Massachusetts Institute of Technology (MIT) in electrical engineering and computer science, where he was a NDSEG Fellow. His B.S. is in electrical and computer engineering from Carnegie Mellon University.

Dr. Alterovitz has worked at Motorola (where he won the Motorola Intellectual Property Award), at IBM, and as a consultant for several national clients. As an invited contributor, he wrote the "Proteomics" section for the *Wiley Encyclopedia of Biomedical Engineering*. Alterovitz has appeared or has been cited for achievements in several national media outlets, including three separate editions of *USA Today*. In 2001, he was selected as one of approximately 20 international delegates to the Canada25 forum (to discuss healthcare/technology) covered by CBC radio, a national TV special, and Canada's *Maclean's*.



Marco F. Ramoni, Ph.D.

Marco F. Ramoni is an assistant professor of pediatrics and medicine at Harvard Medical School and an assistant professor of health sciences and technology at the Harvard University and the Massachusetts Institute of Technology Division of Health Sciences and Technology. He is also the associate director of bioinformatics at the Harvard Partners Center for Genetics and

Genomics and the director of the National Library of Medicine Training Fellowship in Biomedical Informatics at Children's Hospital Boston. He is also the director of the course "Biomedical Informatics" at the Harvard-MIT

Division of Health Sciences and Technology, core faculty of the course “Genomic Medicine” at Harvard Medical School and a member of the curriculum committee of the Cellular and Molecular Medicine track of the Medical Physics and Medical Engineering graduate program at Harvard-MIT Division of Health Sciences and Technology. He is the cofounder of Bayesware LLC, a software company developing machine-learning programs based on Bayesian methods. He received a Ph.D. in biomedical engineering and a B.A. in philosophy (epistemology) from the University of Pavia (Italy) and his postdoctoral training from McGill University, Montreal (Canada). He held academic and visiting positions at the University of Massachusetts, the University of London (United Kingdom), the Knowledge Media Institute (United Kingdom), and the University of Geneva (Switzerland). He is author of over 90 publications in genetics, biomedical informatics, statistics, and artificial intelligence.

Index

- β -carotene, 275
- 1,3-Propanediol case study, 277
- 1/f property, 67–68
- ABI analysis software, 67
- Abstraction barrier, 245
- Acylated homoserine lactones (AHL), 168–69, 237
- Adjacency matrices, 196
- Affymetrix, 298, 307
- A-I mRNA editing, 292
- Alcaligenes eutrophus*, 266
- Alignment algorithm, 114
- Allele
 - common variants, 216
 - major, 206, 207
 - minor, 206, 207
 - wild, 206
- Allosteric aptamer, 235
- Alu repeats, 291
- Amino acids, 11
- Amorphadiene synthase gene (ADS), 276
- Amplitude modulation, 92, 93
- Antialias filters, 106, 108
- Anticodons, 9
- Antinotch filters, 70
- Antisense RNA, 13
- Apoptosis
 - defined, 86
 - system identification, 86–89
 - TNF-EGF-insulin, 86
- Applied cellular engineering, 263–78
- Aptamers, 290
- Asynchronous schedule, 298
- ATP (adenosine triphosphate), 3
- Attractors, 131
- Augmented Network (AN), 220
- Automated Clone Evaluation (ACE) software tool,
 - 33–34
 - defined, 33
 - workflow, 34
- Automated mining, 24–25
- Background correction, 109–12
 - algorithm, 109–11
 - baseline extraction, 111–12
- Backtracking algorithm, 306
- Bacterial chemotaxis, 342–44
 - run, 342, 343
 - tumble, 342
- Bacterial colony, 20
- Bacterial culture, 20
- Bacterial transformation, 20
- Balanced binary code, 308–9
- Bandpass filters, 92
- Barcode labels, 35
- Basecalling, 66, 93
- Baseline
 - adjusting for spectra, 111
 - correction, 65–66, 111
 - estimated, subtracted, 112
- Base schedule, 298, 299, 307–8
- Bayesian computations, 209
- Bayesian model selection, 214
- Bayesian networks (BNs), 143, 210–20
 - applications, 221–24
 - classifiers, 219–20
 - cross-validation techniques, 219
 - defined, 210
 - displaying structure dependency, 222
 - for genetic analysis, 205–25
 - modularity, 213
 - modular representation, 212
 - as multivariate dependency model, 210
 - probabilistic reasoning with, 212
 - reasoning, 217–19
 - risk prediction, 219–20
 - strategies for learning, 213
 - structure example, 211
 - validation and inference, 219
- Bayes score, 216
- Bayes theorem, 216, 218
- Bernoulli random variables, 344
- Biased random walk, 342
 - multinode biased, 347–49
 - single-node, 346–47
 - trajectory, 355
 - See also* Random walk
- Bifan motif, 190
- Bifurcation analysis, 162–64, 171
- Binary parallel divider
 - construction of, 331–34
 - defined, 331
 - See also* Two large prime numbers
- Binary parallel subtractor, construction, 330–31

- BioBricks, 231
 - assembly, 250
 - construction fundamentals, 243–46
 - ends, 244
 - matching, 249
 - parts, 244, 246–48
 - prefixes, 244
 - registry, 251
 - standard, 244
 - suffixes, 244
- Biochemical logic circuits, 231
- Biochemical reactions
 - collection, 89
 - network modeling, 90
 - system analysis, 89–92
- Bioinformatics Toolbox, 107, 121
- BioJADE, 238, 243–61
 - architecture, 248–51
 - aspects, 248–49
 - Basic package, 248
 - compilation of Repressilator design, 253, 254
 - data sheets, 259, 260
 - D-FLUX, 254
 - DNA aspect, 248, 250
 - functional network, 248
 - fundamentals, 243–46
 - icon aspect, 251
 - in iGEM competitions, 259
 - incompatible parts, 252
 - inspiration, 243–44
 - measurement, 259–61
 - part repositories, 251
 - priorities, 258
 - reality check, 257–58
 - registries, 251
 - schematic, 248, 249–50
 - simulation aspect, 249
 - simulation generation, 256–57
 - simulations, 254–57, 258–59
 - star schema, 251
 - Stochastirator, 255, 256
 - in Synthetic Biology Summer Design Competition, 259
 - Tabasco, 255–56
 - using, 251–54
- Biological circuits, 232–36
 - components, 231
 - design, 257
 - early, 232
 - feedback loops, 235–36
 - genes, 232
 - logic gates, 236
 - oscillators, 236
 - riboregulators, 234–35
 - toggle switches, 236
 - transcriptional cascades, 232–33
- Biological computing, 320
- Biological inverter, 233
 - defined, 233
 - kinetics, 234
- Biological molecules, 102
- Biological signal processing, 49–94
 - at cell level, 49–94
 - concepts, 51–58
 - overview, 49–51
 - signal detection and estimation, 59–74
 - system identification and analysis, 74–93
 - See also* Signal processing
- Biological systems
 - defined, 26
 - engineering, 263–65
 - modularity, 263
 - promoter, 264
- Biomarkers, 101
- Biosynthetic pathways, 267
- Bistability, 236
- Black-box algorithms, 105
- Black-box diagram, 246
- Blind deconvolution, 63
- Blurring function, 63
- Boolean networks, 128, 138, 139–43
 - algorithm REVEAL, 141–43
 - entropy and mutual information, 140–41
 - genes as binary (ON/OFF) variable, 139
 - wiring diagram, 139
- Bulge loops, 288
- Calcium oscillations, 92
- Calibration, 109
- Capillary electrophoresis, 101
- Capping, 291
- Catalytic RNAs, 290
- cDNA microarrays, 135, 136
 - introduction, 135
 - schematic representation, 136
- Cell-based assays, 36, 39–42
 - challenges, 40
 - ectopic expression, 40
 - highly parallel approach, 41
 - informatics support, 40–42
 - pooled approach, 41
 - readouts, 40
- Cells
 - as controlled environment, 128
 - as dynamical system, 133
 - functions, 3–4
 - information handling, 4–5
 - as input/output (IO) system, 127
 - manufacturing plant functional parallels, 4
 - robustness, 128, 133
 - structure, 3
 - subsystems, 127
 - See also* DNA
- Cell screening, 40
- Cellular catalytic machinery, 265
- Cellular engineering, 263–78
 - biological systems, 263–65
 - case study, 277

- cellular catalytic machinery, 265
- early successes, 265–66
- experimental methods, 271–77
- frontiers, 277–78
- network models and analysis, 266–71
- post-transcriptional control strategies, 274–76
- regulation, 270
- tools, 266–77
- transcriptional control strategies, 272–74
- translational control strategies, 276–77
- Cellular networks, 181–89
 - adjacency matrices, 196
 - case studies, 164–71
 - genetic regulation, 182–84
 - kinetic models, 153–64
 - metabolic, 192–95
 - metabolic regulation, 185–86
 - methodologies, 153
 - modeling, 151–72
 - motifs, 189–91
 - in predicting essential genes, 195
 - protein, 191–92
 - protein-protein interaction, 184–85
 - reachability, 197–98
 - scale-free, 186–89
 - topological analysis, 181–201
 - topology, 189–98
- Cellular noise, 158–61
- Central Dogma, 5, 289–90
- Chemical localization, 363
- Chemical master equation (CME), 159
- Chemical noise, 109
- Chemical tracking, 341
- Chemoreceptor clustering, 355
- Chemoreceptor cooperation
 - exploitation, 359
 - model, 342
- Chemotaxis, 341–63
 - algorithms for diffusive environments, 345–60
 - algorithms performance comparison, 360–61
 - bacterial, 342–44
 - behavior, 355
 - defined, 342
 - multichemoreceptor cooperation, 350–60
 - multinode biased random walks, 347–49
 - random walk, 341, 344–45
 - receptors, 343
 - run, 342, 343
 - single-node biased random walk, 346–47
 - summary, 361–63
 - tumble, 342
- Cholesteryl ester transfer protein (CETP), 222, 223
- Classifiers, 219–20
 - BN as, 219
 - examples, 220
 - Naive Bayes, 220
 - predictive accuracy, 220
- Clone production, 19, 25–31
 - automation, 27–31
 - challenges, 25
 - gene amplification, capture, isolation, and sequencing, 27
 - high-throughput, 29, 31
 - informatics, 31
 - pipeline, 27–28
- Clones
 - defined, 20
 - distribution, 34–35
 - in functional proteomics approaches, 35–42
 - human protein-coded, 36
 - information, quality control, 35
 - maintenance, 34–35
- Cloning project
 - reference sequences assembly, 25
 - target gene selection, 21–25
- Closed-loop mode, 128, 130
- Clustering coefficients, 193, 194
- Codes. *See* DNA codes
- Codewords, 301
 - binary representation, 305
 - free-energy tables, 306
 - joint constraints, 304
 - nucleic weight, 305–6
 - reversible, 304
 - secondary structure, 303
 - See also* DNA codes
- Coding
 - constraints, 304
 - error-control, 307
 - quality-control, 308
 - theory, 308
 - See also* DNA codes
- Colony isolation, 30
- Colony selection robotics, 30
- Color filter matrix estimation, 66
- Color matching, 297
- Common disease, 216
- Comparative genomic hybridization (CGH)
 - microarrays, 135
- Complementary DNA (cDNA), 298
- Complexity, 16
- Complex trait, 210
 - defined, 210
 - network representation, 221–24
- Component-wise Boolean OR function, 309
- Computational biology, 72
- Conditional entropy, 140–41
- Conditional probability tables, 213
- Continuous-time systems, 55
- Continuous variables, 52
- Control and systems theory, 127–47
 - concepts, 128–33
 - in systems biology, 133–35
- Controlled variable system, 129
- Control system
 - closed-loop mode, 128, 130
 - open-loop mode, 128, 129

- Control theory, 133–35
- Control variable, 129
- Cross-spectral properties, 72–73
- Cross-talk, 75
- Cross-validation techniques, 219
- Cruciforms, 288
- Cryptographic systems, 299–300
- Cutoff, 188
 - natural, 189
 - for node degrees, 188
- Cyclic core, 305

- Dangling strands, 288
- Data acquisition methods, 102
- Data capture, 37
- Data preprocessing, 104–5
- Date hubs, 197
- Decay, 234, 291
- Degree of connectivity, 286
- Denosing, 65
- DES (Data Encryption Standard), 300
- Deterministic dynamical system, 132
- D-FLUX, 254
- Dhariwal algorithm, 361
- Diffusive field modeling, 351
- Dimerization, 234
- Directed acyclic graph (DAG), 210, 211
- Directed evolution, 238, 239
- Directed stochastic dependencies, 210
- Directed traveling salesman (DTS) problem, 294
- Dirichlet distributions, 214
- Discrete Fourier transform (DFT), 52, 53, 73
- Discrete motifs, 264
- Discrete-time dynamical systems, 132–33
- Discrete-time systems, 56
- Discrete variables, 52
- Discrete wavelet transform (DWT), 122
- Discretization algorithms, 217
- Dissociation, 234
- DNA, 3
 - background, 321–23
 - basecalling, 66, 93
 - binding proteins, 245
 - bound, 77
 - chip manufacturing, 298
 - cryptography, 299–300
 - data encryption, 285
 - defined, 321
 - double-helix formation, 293
 - editing, 310
 - folding, 303
 - genetic information, 4
 - microsatellite, 293
 - model comparison, 322–23
 - nanoparticle assembly, 300–301
 - oligosequences, 298
 - one-time pads, 299
 - proofreading, 291–92, 293
 - purification, 20
 - RBS, 247
 - replication, 6
 - schematic representation, 7
 - self-hybridized, 285
 - sticker-based model, 323
 - tiles, 297
 - tubes, 321–22
- DNA-based algorithms, 320
- DNA-based computing, 322
- DNA codes, 301–7
 - balanced, 309
 - from complex Hadamard matrices, 305
 - constant-composition, 305
 - construction, 304–5
 - design, 286
 - design starting point, 285
 - structures, 305
 - superimposed, 285
- DNA computers, 294–97
 - alternative, 295
 - for breaking down cryptographic systems, 300
 - combinatorial questions addressed on, 295
 - defined, 294
 - operations of universal Turing machine, 297
- DNA copy (cDNA), 136
- DNA microarrays, 15–17, 23, 78, 298–99, 307–10
 - asynchronous schedule, 298
 - base schedule, 298, 307–8
 - cDNA, 135, 136
 - fabrication and testing process, 309
 - hybridization, 136
 - linkers, 298
 - manufacturing process, 307
 - mask, 298
 - mask design, 308
 - multiplexed, 309
 - oligonucleotide, 135
 - operational principle, 285
 - opportunity, 137
 - principle, 136
 - production failure detection, 307
 - quality control, 308
 - synchronous schedule, 298
- DNA polymerase
 - defined, 6
 - DNA replication by, 8
- DNA sequencing, 50, 60–67
 - algorithms, 67
 - detailed signal processing model, 66
 - homomorphic blind deconvolution, 63–64
 - mapping, 61
 - signal processing model, 61
 - signal/system definition, 60–61
 - See also* Sequences
- Double-stranded RNA (dsRNA), 291
- Down sampling, 106, 107–9
- Duplication-divergence model, 187, 188
- Dyck's paths, 311

- Dynamical systems
 - cells as, 133
 - control theory and, 129
 - DBNs for, 143
 - defined, 129
 - deterministic, 132
 - discrete-time, 132–33
 - linear, 131
 - nonlinear, 131
 - stability, 131
 - stochastic, 132
- Dynamic Bayesian networks, 128, 133, 143–46
 - algorithm, 145, 146
 - defined, 143
 - for dynamical systems, 143
 - Kalman filter representation, 144
 - linear Gaussian, 144–46
 - performance, 144
 - true parents, 146
- Dynamic model, 138
- EcoCyc database, 181, 184
- E. coli*, 181, 195
 - 1,3-propanediol production in, 277
 - enzymes, 265
 - essential genes, 200
 - genetic regulation network, 183, 199
 - grown in glucose, 186
 - metabolic network, 200
 - PPI graph, 184, 199
 - Repressilator, 236
 - transcriptional regulation network, 190
- Edge weights, 186
- Editing
 - A-I mRNA, 292
 - alternative forms, 291
 - defined, 291
 - DNA, 291
 - gRNA, 292
 - insertion and deletion, 292
 - RNA, 291–93
- Electronic nose designs, 362
- Electron-ion interaction potential (EIIP), 72
- Electron Ionization (EI), 102
- Electron multiplier, 102
- Electrospray Ionization (ESI), 102, 103
- Electrospray Ionization Mass Spectrometry (ESI-MS), 102
- Elowitz Repressilator. *See* Repressilator
- Emanuel syndrome, 293
 - palindromic repeats, 294
 - translocations, 294–95
- Entrez Gene IDs, 35, 42
- Entropy, 140–41
 - conditional, 140–41
 - joint, 140
- Enzymatic chemical reactions, 268
- Enzymes, 265
- Epidermal growth factor (EGF), 86
- Epidermal growth factor receptor (EGFR), 152
- Erdős-Rényi model, 187, 193
- Essential genes, 195
 - biological testing for, 195
 - defined, 195
 - detection, 201
 - functional clustering, 198–200
 - graphs, 200
 - integration in essential gene detection, 201
 - topology, 199–200
 - See also* Genes
- Eukaryotic genes, 22
 - model organism, 181
 - set selection, 22
 - See also* Genes
- Exceptional longevity (EL), 210
- Exons, 290
- Experimental sequences, 32–33
- Expressed sequence tags (EST), 77
- External stability, 131
- Factoring
 - difficulty, 336
 - integers, 335
 - use of, 337
- Factorization, of posterior probability, 215
- Farnesyl pyrophosphate (FPP), 276
- Fast Atom Bombardment (FAB), 102
- Feedback
 - balance, 134
 - control, 128–29
 - control mechanisms, 155
 - defined, 128
 - modules, 155
 - negative, 129, 132, 134, 155, 235
 - positive, 129, 134, 155, 235–36
 - promotion, 270
 - regulation structures, 133
 - repression, 270
 - sensor, 129
- Feedback loops, 234, 235–36
 - in biological circuits, 235
 - bistability, 236
 - defined, 235
- Feed-forward loops, 190
- Fick's Second Law, 351
- Finite Impulse Response (FIR) filter, 106
- First-order reversible chemical reaction, 91
- FLEX Gene LIMS, 31
- Flux, 267, 275
 - analysis, 267–69
 - basic models, 267
 - predictions, 271
- Flux balance analysis (FBA) method, 186
- Folding, 288, 303
 - DNA, 303
 - Nussinov's algorithm, 306, 307
 - RNA, 312
- Fourier analysis, 83

- Fourier transforms, 52
 - of continuous-time variables, 56
 - disadvantage, 54
 - discrete (DFT), 52, 53, 73
 - frequency information, 54
 - inverse, 63, 64
 - short-time, 54
- Fragile chromosome breakage model, 294
- Fragile X, 293
- Frequency
 - information, 54
 - modulation, 92, 93
- Frequency-domain representation, 52–54
- Friedreich's ataxia disease, 293
- Functional clustering, 199
 - concept characteristics, 199
 - defined, 200
 - See also* Essential genes
- Functional network, BioJADE, 248
- Functional proteomics, 17–18
 - building gene collections for, 18–35
 - clones use, 35–42
 - defined, 18
 - information management, 37
 - See also* Proteomics
- Functional RNA (fRNA), 290
- Gaussian distribution, 110, 144–45, 217
- Gaussian kernel, 113
- Gel electropherograms, 101
- Gel electrophoresis, 20
- GenBank, 25
- Gene expression, 137
 - arrays, 75
 - case study, 164–66
 - control, 11–12, 274
 - defined, 11
 - mathematical representation schemes, 165
 - measurement, 101
 - microarray technology, 77–78
 - profiles, 81
 - quantification, 152
 - signal definition, 78–79
 - single, modeling, 164
 - time series, 141, 143
- Gene identification, 67–71
 - comparison methods, 71
 - DNA sequence analysis and, 70
 - DNA signal processing for, 69–71
 - DNA signal properties, 67–68
- Gene knockout
 - defined, 184
 - radioactive, 195
- Gene networks, 137–46
 - benefits, 137
 - Boolean, 139–43
 - defined, 137–38
 - dynamic Bayesian, 143–46
- Gene Ontology (GO), 24
 - defined, 198
 - donors, 200
 - enrichment, 198, 199, 200
- Generalized Hadamard matrix, 305
- General System Theory, 49
- Gene regulation systems, 77–84
 - gene expression microarray technology, 77–78
 - gene expression signal definition, 78–79
 - identification, 79–84
 - modeling, 157
- Gene regulatory network (GRN), 302
- Genes
 - amplification, 27
 - biological circuits, 232
 - candidate, 205
 - clone set production, 19
 - cloning into plasmid vectors, 26
 - clustering, 138
 - collections, building, 18–35
 - defined, 182
 - essential, 195
 - eukaryotic, 22
 - exons, 290
 - expression level, 290
 - GO enriched, 198
 - introns, 290
 - in operon, 275
 - Pseudomonas*, 31
 - regulation, 264
 - regulatory networks, 77
 - silencing, 291
 - target, 21–25
 - transcriptional responses, 79
- Genetic code, 11
- Genetic engineering, 12–13
- Genetic logic circuit, 237
- Genetic regulation networks, 182–84
 - binding of RNA polymerase (RNAP), 182
 - detection rate, 196
 - E. coli*, 183
 - gene knockout, 184
 - interactions, 183
 - network motifs, 189–91
- Genetic toggle switch, 231
- Genome
 - coverage, 22
 - genotyping, 206
- Genomic research, 17
- Genotypes, 216
- Genotyping
 - dependency structure, 208
 - errors, 208
 - genome-wide, 206
- Gibbs sampling, 218
- Gillespie algorithm, 159, 160
- Global Markov property, 212
- Global precision, 214
- Glucose binding proteins, 274

- Glycerol stock, 20
Gradient sensing, 342
Gram-Schmidt orthogonalization, 80
Graph theory, 192
Green fluorescent protein (GFP), 13
Guide RNA (gRNA), 292
- Hadamard matrices, 305
Hairpin loops, 288
Hamming distance, 302, 303
 constraints, 304
 guarantee, 309
 minimum, 302
 reverse-complement, 302, 304
 use, 303
Hamming weights, 305
HapMap Project, 206
Harvard Institute of Proteomics (HIP), 27, 30
Heat map, 114
 alignment, 115
 misalignment, 115
Heat shock proteins, 134
Hebbian learning, 350–51
 algorithm, 352, 353
 algorithm diagram, 353
Hebb's rule, 350
Hidden Markov models, 70, 133
Hierarchical modularity, 193, 194
High-density lipoprotein (HDL) cholesterol, 221, 223
High-frequency components, 119
High resolution, 107
High-throughput clone production, 29, 31
High-throughput data, 135
High-throughput protein production, 36–38
Hill coefficient, 157
Hill kinetics, 157
Homomorphic blind deconvolution, 63–64
Homomorphic signal processing, 63
Horton-Strahler (HS) number, 310, 311
Hubs, 196–97
 benefit, 197
 date, 197
 dynamical position, 197
 party, 197
 reconciling modules with, 195
 See also Modules
Human APOB gene, 292
Human Genome Project, 101, 135, 205
Huntington disease, 293
Hybridization, 136
 coding-theoretic view, 301–13
 defined, 285, 287
 experiments, 286
 imperfect, 288
 introduction, 286–89
 self-, 285
 sequence, 285–313
 WC rules, 301
- Icon aspect, 251
Indigo, 266
Inducible promoter systems, 272
Infinite impulse response (IIR), 70
Informatics support
 cell-based assays, 40–42
 clone production, 31
 high-throughput protein production, 38
 protein array analysis, 39
 sequence analysis, 33–34
Input/output system (IO), cell, 127
Input variables, 129, 130
Insulin, 86
Integers
 factoring, 335
 unsigned, 323
Interaction network, 138
Interference, 75
Internal branching, 288
Introns, 290
Inverse Fourier transform, 63, 64
Inverted repeats, 291, 301
Ion intensity, 118, 122
Ionization, 103
 electron, 102
 electrospray, 102
 matrix-assisted laser desorption, 102
 soft, 102
 soft laser desorption, 102
 techniques history, 102–3
Ion peaks, 117
 identifying, 122
 resolution between, 121
- Jacobian matrix, 163
Jacobsen syndrome, 294
Joint entropy, 140
- K2 algorithm, 216, 217
Kadar/Virk algorithm, 361
Kalman filter, 133, 143
 DBN representation, 144
 defined, 133
Key recovery, 300
Kinetic models, 147
 construction and analysis, 153–64
 modeling resources, 153–54
 modular formulation, 154–56
 ODE-based, 158
 parameter estimation, 153–54
Kinetic rate equations, 232
Kinetics, 267
 basic, 156–57
 biological inverter, 234
 Hill, 157
 Michaelis-Menten, 91, 156, 162, 166
 synthetic biology, 240
Knowledge discovery techniques, 209

- Laboratory information management system (LIMS), 19, 27, 31
- Law of Mass Action, 90
- Levenshtein distance, 303
- LifeTrace algorithm, 67
- Linear and time-invariant (LTI) filtering, 61, 62
- Linear and time-invariant (LTI) systems, 55–56
 - continuous-time, 56
 - defined, 55
 - discrete-time, 56
 - eigenfunctions, 56
 - shift invariance properties, 56
- Linear dynamical systems, 131
- Linear fit, 120
- Linear Gaussian networks, 144–46
 - Gaussian distribution, 144–45
 - marginal likelihood, 145
- Linear regression, 120
- Linkage disequilibrium (LD), 207
- Linkers, 298
- Liquid chromatography mass spectrometry (LC-MS), 101
- Localization time, 358, 359, 360
- Local Markov property, 212, 215
- Logarithmic sensitivity, 162
- Logic gates, 236
- Logistic regression models, 208, 209
- Loop-to-loop interactions, 288
- Lorentzian point spread function, 63
- Low-density lipoprotein (LDL) cholesterol, 221, 223
- Lowess filter smoothing, 120
- Low-resolution spectra, 116
- L-threonine, 185
- Lycopene, 273

- Major allele, 206, 207
- Marginal likelihood, 145, 214
- Markers
 - biomarkers, 101
 - SNPs as, 207
 - X-linked, 212
- Markov blanket, 212, 218
- Markov chains, 57, 90
- Markov property
 - global, 212
 - local, 212, 215
- Masks, 298
 - design, 308
 - Gray code, 308
- Mass analyzers, 103
- Mass/charge, 104
 - quadratic equation constants, 104
 - values, aligning, 112–15
- Mass spectrometer, 112
- Mass spectrometry (MS)
 - data, 109
 - data acquisition methods, 102
 - data processing, 104–5
 - detection of ions, 104
 - example data, 105
 - ionization, 103
 - ionization techniques, 102–3
 - liquid chromatography (LC-MS), 101
 - sample preparation, 103
 - separation of ions, 103–4
 - signal processing methods, 101–22
- Mass spectrum, 104
- Mass transfer coefficient, 268
- Mathematical modeling
 - challenges, 152
 - uses, 151–52
- MATLAB, 196
- Matrix-Assisted Laser Desorption Ionization (MALDI), 102, 103
- Matrix-Assisted Laser Desorption Ionization Mass Spectrometry (MALDI-MS), 102, 104
- MAYA, 296
- Median steps (MS), 360
- MegaBACE sequencers, 67
- Membrane support, 267–68
- Mendelian disease, 205
 - genetic basis, 206
 - recessive, 206
- Messenger RNA (mRNA), 7, 10, 164
 - codon, 10
 - expression levels, 17
 - gene information transfer to, 289
 - molecule, 182
 - nucleotides, 10
 - pre-, 290
 - stability, 276
 - transcripts, 17
- Metabolic engineering, 266
- Metabolic networks
 - clustering coefficients, 193, 194
 - comparison, 194
 - diameter, 193
 - modularity, 193
 - scale-free structure, 192
 - simplicity, 192
 - small chemical compound transformation, 192
 - topology, 192–95
- Metabolic regulation, 185–86
- Metabolites
 - conversion, 266
 - network, 269
 - overproduction, 267, 269
 - regulation, 270
 - toxicity, 269
- Michaelis-Menten constants, 167, 168, 268
 - ratio, 167
 - total protein concentration and, 167–68
- Michaelis-Menten kinetics, 91, 156, 162, 166, 268
- MicroRNA (miRNA), 290
- Microsatellite DNA, 293
- Minimum Description Length (MDL), 217

- Mining
 - automated, 24–25
 - published reports, 24
- Minor allele, 206, 207
- Mitogen-activated protein kinase (MAPK), 151
 - models, 152
 - parameters, 167
 - pathways, 152
- Mobility shift correction, 66
- Model-based estimation, 65–67
- Modularity
 - Bayesian networks (BNs), 213
 - biological systems, 263
 - defined, 191
 - hierarchical, 193, 194
 - interpretations, 194
 - in large model search, 216
 - metabolic networks, 193
- Modular search, computational gain, 215
- Modules
 - defined, 191
 - local, 197
 - nodes, 194
 - reconciling with hubs, 195
- Molecular algorithm, 322
- Monte Carlo method, 159
- Moralization, 224
- Motifs
 - bifan, 190
 - directionality of regulation interactions, 191
 - discrete, 264
 - generalizations, 190
 - in genetic regulation networks, 189–91
 - small-scale, 189–90
- Motor pRNA of bacteriophage, 301
- Motzkin and Schroeder paths, 312
- Multicellular systems, 236–38
- Multichemoreceptor cooperation
 - algorithm, 351–60
 - diffusive field, 351
 - for gradient tracking, 350–60
 - Hebbian learning, 350–51
- Multidimensional signals, 54
- Multinode biased random walks, 347–49
 - robot mean free path (MFP), 347
 - run-and-tumble strategy, 348
 - for source tracking, 347–49
- Multiplexed arrays, 309
- Multiplexing matrix, 309, 310
- Multistage filters, 70
- Multivariate statistics models, 208
- Mutagenesis, 239
 - random, 239–40
 - targeted, 239–40
- Mutual information, 141
- Naive Bayes classifier, 220
- Natural cutoff, 189
- NCBI Entrez Gene project, 42
- Negative feedback, 129, 132, 134, 155, 235
- Network motifs, 189–91
- Network topology, 181
- Noise, 64
 - chemical, 109
 - chemotaxis implementation, 359
 - filtering, 66, 76
 - reduction by filtering, 119
 - smoothing, 119–21
 - standard deviation, 346
- Nonattacking knights problem, 295
- Nonlinear dynamical systems, 131
- Nonlinear systems theory, 56–57
- Nonsense-mediated decay, 291
- Normalization, 101
 - errors, eliminating, 118
 - to highest ion peak, 117
 - mass spectra, 116
 - relative intensity, 116–19
 - robust, 116
- No-sharp-turn constraint, 312
- Nussinov's folding algorithm, 306, 307
- Oligonucleotide microarrays, 135
- Oligonucleotide primer, 2
- Oligonucleotide sequences, 289
- Open-loop mode, 128
- Operons, 275
- Ordinary differential equation (ODE), 158
- Organism-specific database IDs, 35
- Oscillator modules, 156
- Oscillators, 234, 236
- Output transformation, 130
- Output variables, 129
- Palindromic repeats, 294
- Parallel comparator, construction, 325–27
- Parallelism, 320
- Parallel one-bit subtractor
 - algorithm, 328–29
 - construction, 327–30
 - defined, 327
 - truth table, 328
 - See also* Two large prime numbers
- Parameter estimation, 153–54
- Parametric sensitivity analysis, 161–62
- Partial least-squares regression (PLSR), 89
- Partitioning techniques, 209
- Parts, 259
 - data model, 247–48
 - encapsulation, 247
 - incompatible, 252
 - repositories, 251
 - representing, 246–48
 - See also* BioBricks
- Party hubs, 197
- Pathway databases, 153

- Peak detection algorithms, 119
- Peak height, 117, 121
- Periodogram averaging, 58
- Perturbations, growth rate, 163
- Petri net formalism, 90
- Phenotypes, 40, 219
- Phosphorylation-dephosphorylation cycle, 166–68
- Phred, 67
- PlasmID, 35
- Plasmid vector, 20
- Poisson distribution networks, 187
- Polyhydroxybutyrate (PHB), 266
- Polyketide synthase (PKS), 278
- Polymerase chain reaction (PCR), 12
 - for averaging mutations, 239
 - defined, 20
- Polymerase per second (PoPS), 245
- Polynomial filtering. *See* Smoothing
- Polynomial fitting, 120
- Polypeptides, 4, 5
- Positive feedback, 129, 134, 155, 235–36
- Posterior probability, 214, 215
 - factorization, 215
 - network model, 217
- Post-transcriptional control strategies, 274–76
- Post-transcriptional gene silencing (PTGS), 291
- Power-law degree distribution, 189
- Power-law distribution, 186, 192
- Power spectral density (PSD), 58, 83
- Power spectrum, 58, 67, 68
- Preferential attachment model, 186, 187
- Pre-mRNA, 290
- Principal component analysis (PCA), 79
 - component analysis with, 87
 - obtaining, 79
 - principal components, 80
- Prior distribution, 213, 214
- Probabilistic model, 110
- Probabilistic reasoning, 217, 219
- Probe neighborhood graphs, 308
- Probes, 136
- Prognostic reasoning, 217
- Programmed cell death. *See* Apoptosis
- Promoters, 7, 246
 - biological module, 264
 - inducible, 272
 - library, 273
 - polymerase generation, 256
- Protein arrays, 36, 38–39
 - feature mapping, 39
 - high-throughput analyses facilitation, 38–39
 - informatics support, 39
 - self-assembling, 38–39
 - signal above ground, 39
 - types, 38
- Protein hotspots
 - defined, 72
 - identification, 71–74
 - prediction of, 73–74
- Protein networks, 55, 191–92
- Protein patterns, 101
- Protein production, 15
 - in functional proteomics approach, 38
 - high-throughput, 36–38
- Protein-protein interaction (PPI) networks, 184–85
 - for cell communication, 184
 - centrally located proteins, 196
 - degree distribution, 188
 - graph, 184, 185
 - hubs position, 197
 - information, 191
 - modules, 191
 - network topology, 191
 - See also* Cellular networks
- Proteins
 - characteristic frequencies, 73
 - cross-spectral properties, 72–73
 - diversity, 5–6
 - DNA binding, 245
 - folding problem, 72
 - functional states, 17
 - heat shock, 134
 - importance, 5–6
 - as pathways, 198
 - as polymers, 6
 - purified, 36
 - regulatory, 264
 - repressors, 12
 - transcription factors, 12
 - transcription/translation into proteins, 11
 - transmembrane (TM), 23
 - unstable, 38
- Protein signaling
 - definition, 85
 - networks, 89
 - occurrence, 84
 - systems, 84–93
 - temporal measurements, 85
- Proteomes
 - defined, 182
 - defining, 15–18
 - understanding, 42
- Proteomic data, 60
- Proteomics, 137
 - defined, 17
 - functional, 17–18
 - research, 17
- Pseudoknots, 288, 303
- Pseudomonas* genes, 31
- Pseudomonas* Genome Project, 22
- Pseudomonas putida*, 266
- Purines, 286–87
- Pyrimidines, 286, 287
- Quadratic fit, 120
- Quality control
 - of clone information, 35
 - coding, 308

- probes, 309
 - at sample level, 34
 - spots, 308, 309
- Quantile, 110
- Random errors, 193
- Random mutagenesis, 239–40
- Random networks, 193
- Random processes
 - defined, 57
 - spectral analysis and, 57–58
- Random walk
 - 2-D, 345
 - biased, 342, 346
 - defined, 344
 - mathematical description, 344–45
 - multinode biased, 347–49
 - multiple biased, 342
 - single-node biased, 346–47
 - single-sensor biased, 341
 - See also* Chemotaxis
- Running Digital Sum (RDS) graph, 311–12
- Reachability, 197–98
 - future modifications, 198
 - graph, 197
 - graph nodes, 198
 - index, 197
 - matrix, 197, 198
- Reasoning
 - Bayesian networks, 217–19
 - probabilistic, 217, 218
 - prognostic, 217
- Receptor cooperation, 346, 361
- Recombination, 239, 240
- Recombinational cloning, 20
- Red fluorescent protein (RFP), 258
- Re-encoding, 291–93
- Reference sequences, 32–33
- RefSeq, 25
- Registry of Standard Biological Parts, 231
- Regulatory networks, 77
- Relational database management system (RDBMS), 251
- Relative intensity
 - normalizing, 116–19
 - rescaling spectra and, 117
- Repressilator, 251–54
 - building, 252
 - in functional network view, 252
 - implementation, 251
- Repressors, 12
- Resampling, 105–9, 113
 - advantages, 105–6
 - algorithm explanation/discussion, 106–7
 - before/after, 107, 108
 - defined, 105
 - disadvantage, 106
 - down sampling example, 107–9
- Resonant recognition model (RRM), 72
- REVEAL algorithm, 141–43
- Reverse-complement Hamming distance, 302, 303
- Reverse engineering cellular networks, 128, 135–37
 - high-throughput data, 135
 - microarrays, 135–37
 - See also* Cellular networks
- Reverse transcribing, 136
- Rhodobacter capsulatus*, 275
- Ribosomal RNA (rRNA), 9, 182
- Ribosome, 9, 10
- Ribosome binding site (RBS), 246–47
- Riboswitches, 275–76, 290
- Ribozymes, 290
- RNA, 5, 6
 - editing, 291–93
 - folding, 312
 - functional molecules, 289–90
 - motif enumeration, 310–13
 - nanoparticle assembly, 300–301
 - planar and tertiary structures, 285
 - role, 290
 - secondary structures, 289
 - self-hybridized, 285
- RNA-based simulations, 258
- RNA inference (RNAi), 39, 291
- RNA polymerase, 7–8, 156, 232
 - binding of, 182
 - cofactor, 134
 - DNA transcription by, 9
 - gene transcription, 12
 - landing pad, 233
- RNA riboregulators, 231, 234–35
 - capability, 235
 - defined, 234
 - DNA sequence, 235
 - example, 235
- Robustness
 - cells, 128, 133
 - quantifying, 162
 - system, 131
 - tradeoff, 114
- RSA public-key cryptosystem, 319, 320
 - breaking, 335–36
 - defined, 319
 - introduction to, 323
 - in message encryption, 335
- Run, 342, 343
- Run-length constraints, 313
- Saccharomyces cerevisiae*, 181, 188, 195, 276
- Satisfiability problem, 295
- Savitzky and Golay filter smoothing, 121
- Scale-free networks, 186–89
 - biological need, 189
 - cutoff, 188, 189
 - degree of connectivity, 286
 - duplication-divergence model, 187, 188
 - heterogeneous, 187
 - power-law degree distribution, 189

- Scale-free networks (*cont.*)
 - random networks versus, 187
 - See also* Cellular networks
- Schematic design mode, BioJADE, 249–50
- Secondary structures, 287
 - branching complexity, 310
 - codewords, 303
 - constraints, 304
 - exact pairings, 306
 - RNA, 289
- SELEX (Systematic Evolution of Ligands by Exponential Enrichment), 290
- Self-assembling protein arrays, 38–39
- Self-hybridization, 285
 - defined, 287
 - introduction, 286–89
 - patterns in DNA sequences, 285
 - See also* Hybridization
- Sensitivity analysis, 161–62
 - in quantifying robustness, 162
 - use, 161
- Sensor cooperation algorithm, 357, 360
- Sequence analysis, 32–34
 - clone set challenges, 32
 - experimental/reference comparison, 32–33
 - gene identification and, 70
 - informatics support, 33–34
- Sequence contig, 20
- Sequence hybridization, 285–313
 - biological point of view, 289–94
 - coding-theoretic view, 301–13
 - introduction, 286–89
 - technological point of view, 294–301
- Sequences
 - estimation with Wiener filtering, 61–62
 - experimental, 32–33
 - identifier, 61
 - reference, 32–33
 - riboregulator, 235
 - self-hybridization patterns, 285
 - structure, 287
 - See also* DNA sequencing
- Sequence trace, 20
- Shine-Delgarno sequence, 246
- Short-time Fourier transform, 54
- Sickle cell anemia, 206
 - cause, 206
 - stroke risk, 221
- Signal detection, 59–74
- Signal estimation, 59–74
- Signaling networks, 75
- Signaling pathways, 74–75
- Signal processing
 - algorithms, 49
 - in biochemical reaction network analysis, 90
 - biological, 49–94
 - concepts, 51–58
 - for gene identification, 69–71
 - homomorphic, 63–64
 - for mass spectrometry, 101–22
 - signals, 51–54
 - spectral analysis, 57–58
 - systems, 54–57
 - systems, examples, 75
- Signals, 51–54
 - biological, 51
 - definition for DNA sequencing, 60–61
 - discrete-time, 52
 - frequency-domain representation, 52–54
 - mathematical functions, 51
 - multidimensional, 54
 - PSD, 58
 - resampling, 105–9
 - time-domain representation, 51–52
- Signal-to-noise ratio (SNR), 346, 359
 - fixed, 357
 - initial average, 356
 - lowering, 357
 - starting, 356
- Simpson's paradox, 211
- Simulations
 - BioJADE, 254–57
 - future, 258–59
 - generation, 256–57
 - RNA-based, 258
- Single-node biased random walk, 346–47
- Single nucleotide polymorphisms (SNPs), 135, 206
 - dependencies between, 208
 - efficiency, 216
 - linking, 212
 - as markers, 207
 - mutation determination, 207
- Singular value decomposition (SVD), 79, 81
 - applying, 80
 - in biological system analysis, 83
 - component analysis with, 87
 - defined, 79
 - eigenvalues resulting from, 82
- Sliding window, 120
- Small interfering RNA (siRNA), 291
- Small-scale motifs, 189–90
- Smoothing, 101
 - defined, 119
 - example, 121
 - Lowess filter, 120
 - methods, 119
 - Savitzky and Golay filter, 121
- Soft laser desorption ionization (SLDI), 102
- Spectral analysis, 57–58
- Spectrum
 - with estimated baseline subtracted, 112
 - with identified ion peaks, 122
 - low-resolution example, 112
 - mass, 104
 - noisy and smoothed, 121
 - plotting, 111

- realignment, 113
- Splicing, 290
- Spring/mass system, 131
- Stability, 131
 - external, 131
 - improving, 258
 - linear, 162
 - mRNA, 276
 - quantitative measure, 162
 - structural, 131
- State equation, 130
- State response, 131
- State-space representation, 130
- State trajectory, 131
- State variables, 130
- Steady-state analysis, 271
- Stem interactions, 288
- Sticky ends, 297
- Stochastic differential equations (SDEs), 160, 161
- Stochastic dynamical system, 132
- Stochasticity, 238–39
- Stochasticator, 255, 256
- Structural stability, 131
- Subclones, 20
- Subgraph roles, 190
- Substrates, 193
- Subsystems, cell, 127
- Sugar-phosphate backbone, 286, 287
- Superimposed codes, 285
- Superimposed designs, 309
- Supervised discretization algorithms, 217
- Surface-Enhanced Laser Desorption Ionization (SELDI), 105
- Surface Enhanced Laser Desorption Ionization Mass Spectrometry (SELDI-MS), 102, 103
- Synchronous schedule, 298
- Synthetic biology, 155
 - BioJADE, 243–61
 - biological inverter, 233
 - challenges, 238–40
 - defined, 231
 - design fundamentals, 231–41
 - directed evolution, 238, 239
 - kinetics, 240
 - multicellular systems, 236–38
 - overview, 231–32
 - random mutagenesis, 239–40
 - recombination, 239, 240
 - standardization, 238
 - stochasticity, 238–39
 - system interface, 240
 - systems, 243
 - targeted mutagenesis, 239–40
- Synthetic population control circuit, 168–71
- System analysis
 - biochemical reactions, 89–93
 - techniques, 161–64
- System identification, 74–93
 - apoptosis, 86–89
 - gene regulation, 77–84
 - protein signaling, 84–93
- System interface, synthetic biology, 240
- Systems, 54–57
 - continuous-time, 55
 - defined, 54–55
 - definition for DNA sequencing, 60–61
 - dimension, 130
 - discrete-time, 56
 - high-coherence, 84
 - LTI, 55–56
 - nonlinear theory, 56–57
 - robustness, 131
 - types of, 55
- Systems biology, 153
 - control theory in, 133–35
 - defined, 127
 - design principles, 133
 - model selection, 147
- Tabasco, 255–56
 - defined, 255
 - D-FLUX wrapper, 255, 256
 - simulation snapshot, 256
 - See also* BioJADE
- Tandem repeats, 293
- Target, 136
- Targeted mutagenesis, 239–40
- Target genes
 - from annotated bacterial genome sequence, 21–22
 - bioinformatic approaches, 21
 - from curated subgroups of eukaryotic genes, 22
 - information sources, 22–24
 - selection, 21–25
- Tertiary structure, 287
- Text-mining tools, 24
- Threading, 308
- Time-domain representation, 51–52
- Time-of-flight (TOF) tube, 103
- Toggle switches, 236
- Topology
 - cellular networks, 189–98
 - defined, 181
 - essential gene, 199–200
 - metabolic networks, 192–95
 - network, 181
 - PPI networks, 191
 - protein networks, 191–92
- Transcription, 7–9
 - defined, 7, 289–90
 - rates, 234
 - by RNA polymerase, 9
- Transcriptional cascades, 232–33
- Transcriptional control strategies, 272–74
- Transcriptional fusion, 12–13
- Transcription factors, 12
- Transfer RNA (tRNA), 9–10

- Translation, 9–11
 - defined, 9–11, 290
 - process, 10
 - rates, 234
- Translational control strategies, 276–77
- Translational fusion, 13
- Translocations, 293
- Transmembrane (TM) proteins, 23
- Tree Augmented Network (TAN), 220
- Trinitrotoluene (TNT), 274
- Tumble, 342
- Tumor necrosis factor alpha (TNF), 86
- Turing machines, 297
- Two large prime numbers
 - binary parallel divider, 331–34
 - binary parallel subtractor construction, 330–31
 - finding, 334–35
 - parallel comparator construction, 325–27
 - parallel one-bit subtractor, construction, 327–30
 - product, construction, 324–25
 - product, factoring, 323–36
- Unsupervised discretization algorithms, 217
- Vienna secondary structure package, 306
- VLSIPS (Very Large Scale Immobilized Polymer Synthesis) methods, 298, 308
- Warp functions, 113
- Watson-Crick complementation, 287, 291, 293
- Watson-Crick rule, 285
- Wiener filtering, 50, 59
 - DNA sequence estimation with, 61–63
 - to separate signal peaks, 62
- Wild allele, 206
- Wiring diagrams, 139
- X-linked genetic marker, 212
- Zymomonas mobilis, 266